iWay Software
An Information Builders Company

iWay

iWay Server Administration for MVS and VM
Release 5 Version 2.0

DN3501102.0103

Contents

Preface

This documentation describes the iWay Server Administration for MVS and VM, presents general information about data management, and provides specific information about customizing individual data adapters. It is intended for data administrators and others who create and maintain data sources and the Server Catalog.

How This Manual Is Organized

This manual includes the following chapters:

Chapter		Contents
1	Server Profiles	Describes how to configure the products you selected during server configuration.
2	Server Security and Data Access Control	Explains the additional configuration steps required to configure server-supported services.
3	Metadata Services With SQLENGINE SET	Explains the functionality provided by National Language Support and how to configure your server to take advantage of it.
4	Metadata Services in Full-Function and Hub Servers	Explains how to modify your configuration, change global parameters, and add supported services.
5	Resource Governor and Resource Analyzer Administration	Explains Resource Governor/Resource Analyzer operations, how to generate reports based on usage monitoring data, and how to govern requests with the Usage Monitor.
6	Server Configuration File Keywords	Provides details on keywords in the server configuration file.
7	MVS Accounting	Describes the optional facility (SMFNUM), which enables the server to generate SMF records for query level and user-level accounting.
8	Application Namespaces	Provides details on Application Namespaces.

Documentation Conventions

The following conventions apply throughout this manual:

Convention	Description
`THIS TYPEFACE` or `this typeface`	Denotes syntax that you must enter exactly as shown.
`this typeface`	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
`underscore`	Indicates a default setting.
this typeface	Represents a placeholder (or variable) in a text paragraph, a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option you can click or select.
this typeface	Highlights a file name or command in a text paragraph that must be lowercase.
Key + Key	Indicates keys that you must press simultaneously.
`{ }`	Indicates two or three choices; type one of them, not the braces.
`[ ]`	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the parameter in the brackets, not the brackets.
`|`	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
`...`	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis points (…).
`.` `.` `.`	Indicates that there are (or could be) intervening or additional commands.

Related Publications

Visit our World Wide Web site, http://www.iwaysoftware.com, to view a current listing of our publications and to place an order. You can also contact the Publications Order Department at (800) 969-4636.

Customer Support

Do you have questions about iWay Server Administration for MVS and VM?

Call Customer Support Service (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your iWay questions. Our consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code number (*xxxx.xx*) when you call.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our World Wide Web site, http://www.iwaysoftware.com. It connects you to the tracking system and known-problem database at our support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of www.informationbuilders.com also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

To learn about the full range of available support services, ask your iWay Software representative about InfoResponse Online, or call (800) 969-INFO.

Information You Should Have

To help our consultants answer your questions most effectively, be ready to provide the following information when you call:

- Your six-digit site code number (*xxxx.xx*).
- Your iWay Software configuration:
  - The iWay Software version and release.
  - The communications protocol (for example, TCP/IP or LU6.2), including vendor and release.
- The stored procedure (preferably with line numbers) or SQL statements being used in server access.
- The database server release level.
- The database name and release level.
- The Master File and Access File.

- The exact nature of the problem:
  - Are the results or the format incorrect? Are the text or calculations missing or misplaced?
  - The error message and return code, if applicable.
  - Is this related to any other problem?
- Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?
- What release of the operating system are you using? Has it, your security system, communications protocol, or front-end software changed?
- Is this problem reproducible? If so, how?
- Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two data sources, have you tried executing a query containing just the code to access the data source?
- Do you have a trace file?
- How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?

User Feedback

In an effort to produce effective documentation, the Documentation Services staff welcomes any opinion you can offer regarding this manual. Please use the Reader Comments form at the end of this manual to relay suggestions for improving the publication or to alert us to corrections. You can also use the Documentation Feedback form on our Web site, http://www.iwaysoftware.com.

Thank you, in advance, for your comments.

iWay Software Training and Professional Services

Interested in training? Our Education Department offers a wide variety of training courses for iWay Software and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our World Wide Web site (http://www.iwaysoftware.com) or call (800) 969-INFO to speak to an Education Representative.

Interested in technical assistance for your implementation? Our Professional Services department provides expert design, systems architecture, implementation, and project management services for all your business integration projects. For information, visit our World Wide Web site (http://www.iwaysoftware.com).

CHAPTER 1

Server Profiles

Topics:

- Levels of Profile
- Order of Execution for Profiles
- Location of Profiles
- Profile Commands
- Profile Commands Reference

During the server installation/configuration step, the EDASPROF global profile is created containing commands relevant to the server type that you chose during configuration. You can code additional commands, if and when required, in any of the supported server profiles. These topics provide information on commands that change the behavior of the session for a connected user. You can also code other commands, such as data access commands, in the server profiles, but it may impact the time it takes for a user application to connect to the server. This is not the case if you use pooled deployment, since EDASPROF is executed only once, upon first user connection.

Levels of Profile

The server supports three levels of profiles to provide flexibility in designing and running production applications.

- **Global profile**

  The first level of profile, a global profile, is a startup file that is automatically created during the installation and configuration of the server. It contains the default environment settings that are required for the proper operation of the server. The global profile remains in effect throughout a user session. You can customize the global profile to change the default settings supplied during installation and configuration.

- **Group profile**

  A group profile is a file used by the server to specify settings that apply to a server environment, but only for a user of a specific security group. Upon connection to the server by a user, the group profile settings are applied, and remain in effect throughout the server session. Group profiles may contain settings that are for the most part defined by the same set of commands used in a global profile. The group profile can be manually created at any time using any operating system editor. This level of profile is only available if security is ON. In MVS, a user can be a member of many security groups but only one group can be used for profile processing. The group ID that is placed in the security control block (ACEE), by the security package established upon successful user ID validation, is used as the group profile. For Resource Access Control Facility (RACF), it is the default group that is returned.

- **User profile**

  The third level of profile, a user profile, is available based on user ID. A user profile is a file used by the server to specify settings that apply to a server environment, but only for a specific user ID. Upon connection to the server by the user ID, the settings are applied, and remain in effect throughout the session. User profiles may contain settings that are for the most part defined by the same set of commands used in a global profile. You can manually create the user profile at any time using any operating system editor.

Order of Execution for Profiles

The order of execution for server profiles is:

- Global Profile
- Group Profile
- User Profile

Note: If you use DEPLOYMENT=POOLED, the EDASPROF global profile is the only profile that is executed.

If two or more server profiles have the same profile command, the values from the last profile that is executed are in effect for the server session.

Location of Profiles

When you configure your server using one of the provided configuration routines, one or more service blocks are generated in the server's configuration file EDASERVE. In the SERVINIT section of the service block, DYNAM commands will have been included to allocate ddnames EDASPROF and EDAPROF.

- EDASPROF points to a member of *qualif*.EDAPROF.DATA, which is the global profile.
- EDAPROF points to *qualif*.EDAPROF.DATA (without a member name) in which the group level and user-level profiles will be searched for.

By having the allocations for the profiles at the service block level, each service block can be configured differently for different groups of users.

Profile Commands

This section describes commands that you can include in any of the supported server profiles. These commands affect the behavior of the server for the duration of the connected session. You can code additional commands, such as data access commands, in any of the supported server profiles. For more information, see the specific data adapter topics in this manual.

Note: For FOCUS Table Services, the server profiles play the same role as the FOCUS profile.

Profile Command Formats

Each server command takes one of three possible formats. The syntax and an example of these formats follow. Make sure you use the correct format for any server command you use.

Syntax How to Use a Direct SET Command

```
SET command=value
```

where:

command

Is the server command.

value

Is the value you select from the available choices.

Example Using a Direct SET Command

```
SET SQLENGINE=DB2
```

Syntax How to Use the SQL Engine SET Command

```
ENGINE [sqlengine] SET command value
```

where:

`sqlengine`

Indicates the data source. You can omit this parameter value if you previously issued the SET SQLENGINE command.

`command`

Is the server command.

`value`

Is the value you select from the available choices.

Example Using the SQL Engine SET Command

```
ENGINE SQLORA SET OWNERID EDAUSER
```

Syntax How to Use an SQL Translator Command

```
SQL
SET command=value
END
```

where:

`command`

Is the SQL Translator command.

`value`

Is the value you select from the available choices.

Example Using an SQL Translator Command

```
SQL
SET APT=OFF
END
```

Profile Commands Reference

The following sections provide the syntax for the profile commands. Individual data adapter topics include a description of the applicable profile commands for that data adapter.

General Commands

The following section describes the general profile commands.

Syntax How to Control the Buffer Size for BLOB or CLOB Data

```
SET BLIM=buffersize
```

where:

buffersize

Is the buffer size to use for retrieving BLOB or CLOB data. If not specified, it will default to 1K. This setting is provided for tuning purposes.

Syntax How to Enable Continental Decimal Notation

```
SET CDN={ON|OFF}
```

where:

ON

Uses a comma as a decimal place instead of a period.

OFF

Does not use a comma as a decimal place instead of a period. This value is the default.

SPACE

Sets the decimal point as a comma, and the thousands separator as a space.

QUOTE

Sets the decimal point as a comma, and the thousands separator as an apostrophe.

Syntax How to Control Century Dates

The DEFCENT syntax is

```
SET DEFCENT=nn
```

where:

nn

Is 19 unless otherwise specified. This is used with the command SET YRTHRESH and provides a means of interpreting the century.

The YRTHRESH syntax is

```
SET YRTHRESH=nn
```

where:

nn

Is zero unless otherwise specified. This is used with the command SET DEFCENT and provides a means of interpreting the century.

Syntax How to Disable SQL Passthru

```
SET DPT=OFF
```

where:

`OFF`

Disables SQL Passthru mode. Any attempt to set the engine at the query level or in an RPC will produce an error message.

Syntax How to Specify the Search Order for a Stored Procedure

```
SET EXORDER=exorder
```

where:

`exorder`

Specifies the search order used by the server to locate a stored procedure. Possible values are:

`PGM` searches the Program Library (IBICPG) only.

`FEX` searches the Procedure Library (FOCEXEC) only.

`PGM/FEX` searches the Program Library, then the Procedure Library.

`FEX/PGM` searches the Procedure Library, then the Program Library.

Syntax How to Control External Sorting

```
SET EXTSORT={ON|OFF}
```

where:

`ON`

Enables the server to use a dedicated external sorting product to sort reports. This value is the default.

`OFF`

Uses the server's internal sorting procedure to sort all requests.

Syntax How to Specify a Language

```
SET LANGUAGE=language
```

where:

`language`

Specifies a language other than American English (the default) for the server environment. The following values are supported: Dutch, French, German, and Japanese.

Syntax How to Control Server Records Sent to the Client

```
SET SLIM=n
```

where:

n

Is the number of records downloaded to the client at one time. This feature:

- Enables the server to respond more quickly to client cancellation and disconnect requests (EDAQUIT).
- Provides a way to control the number of rows received by the client application for each data fetch issued.
- Allows you to send requests to the same server without having to wait for all rows to be retrieved from the first request, consequently improving response times for all requests.
- Resolves client memory limitation issues by reducing record spooling on the client.

Note:

- The client application can override the server setting by issuing an EDASET function call in the API. This can be set globally for all connections from a particular client application, or locally, for a specific connection. It does not affect other users or application connections.
- This setting is also required for BLOB and CLOB support.

Syntax How to Enable SMARTMODE

```
SET SMARTMODE={ON|OFF}
```

where:

ON

Enables SmartMode operations on requests processed by the server.

OFF

Disables SmartMode operations on requests processed by the server. This value is the default.

Syntax How to Specify the RDBMS to Be Accessed in Direct Passthru Mode

```
SET SQLENGINE=sqlengine
```

where:

sqlengine

Specifies the RDBMS to be accessed in Direct Passthru mode. This can be used only in the global profile. User profiles should *not* contain this command, as this defines single RDBMS access in Direct Passthru mode.

Syntax How to Allow Mixed-Case SQL Statements

```
SET UPCASE={ON|OFF}
```

where:

ON

Before passing the request to the server, the client API will uppercase all SQL statements (but not any data embedded in quotation marks).

OFF

Signifies that the server/database can understand both uppercase and lowercase commands. This value is the default.

SQL Translator Commands

The following sections describe the SQL Translator profile commands.

Syntax How to Set Automatic Passthru

```
SQL
SET APT=apt
END
```

where:

apt

Enables SQL Translation Services to translate eligible incoming requests into native SQL dialects. Possible values are:

ON turns Automatic Passthru on. When this is the value of the APT setting, SQL Translation Services analyzes each inbound request to determine whether Dialect Translation can be invoked to pass the request to SQL Passthru Services. This is the default value for a Hub or Full-Function Server.

OFF turns Automatic Passthru off. All SQL requests are processed using DML Generation, which translates them into the internal Data Manipulation Language to be processed by the appropriate data adapters.

Note: Turning Automatic Passthru OFF can increase the processing overhead incurred.

ONLY indicates that all requests must be appropriate for Dialect Translation, that is, they must meet the requirements defined in the *SQL Reference* manual. No SQL requests go through DML Generation. If a request does not meet the requirements, the query is rejected, and an error is returned to the client application.

This enables data administrators to configure access to servers so as to avoid any overhead that would be incurred when generating DML.

For more information on SQL Translation Services and Automatic Passthru, see the *SQL Reference* manual.

Syntax How to Allow the SET ALL= Command

```
SQL
SET EMITALL={YES|NO}
END
```

where:

YES

Includes SET ALL=OFF as part of the translated request so that any direct SET ALL= command already in effect will be ignored. This situation is only active when a JOINed request is being translated. This value is the default.

NO

Does not include SET ALL=OFF in the translated request so that the SET ALL= command can be used to control short path processing of hierarchical file structures in a JOINed request.

Syntax How to Process a FILLER Field in an SQL Statement

```
SQL
SET FILLER=filler
END
```

where:

filler

Enables you to select how a field name of FILLER will be processed in an SQL request. Possible values are:

ON indicates the field name is valid in an SQL statement BUT is not included in the SELECT * expansion.

OFF indicates that FILLER is ignored. This value is the default.

STAR indicates FILLER is a valid field name and is included in the SELECT * expansion.

Syntax How to Issue the DEFINE FILE CLEAR Statement

```
SQL
SET NODEFCLEAR={ON|OFF}
END
```

where:

ON

Enables the DEFINE FILE *filename* CLEAR statement. This value is the default.

OFF

Disables the DEFINE FILE *filename* CLEAR statement.

Syntax How to Include GROUP Fields in a SELECT * Request

```
SQL
SET SQLTGROUP={ON|OFF}
END
```

where:

ON

Includes GROUP fields in a SELECT * request.

OFF

Does not include GROUP fields in a SELECT * request. This value is the default.

HiperEDA Commands

The following section describes the HiperEDA profile commands. HiperEDA is a file management feature that accelerates server processing. Under MVS, it uses hiperspaces to speed up processing of temporary files and cache memory.

Syntax How to Issue HiperEDA SET Commands

The following is a list of SET commands available for HiperEDA:

Command	Description
SET CACHE={0|*n*}	0 Allocates no pages to cache, in effect turning off the cache facility. This is the default for standard cache. *n* Is the number of pages you wish to allocate to cache. The minimum is two pages. The default for HiperCache is 256 pages (that is, one megabyte). When CACHE is set to any positive value, the cache facility is on.
SET HIPERCACHE=*nnn*	*nnn* Is the number of hipercache (4K) pages. The default is 256 (4K) pages or 1M in size.
SET HIPEREDA={ON|OFF}	ON Activates HiperEDA after HiperEDA has been installed. This value is the default. OFF Deactivates a HiperEDA session. **Note:** If HiperEDA is not installed, the SET HIPEREDA command is disabled. You can determine if HiperEDA is installed and active by issuing the ? SET command in a stored procedure. If HiperEDA is installed, the command's display includes the message HIPEREDA ON or HIPEREDA OFF.
SET HIPEREXTENTS=*nnn*	*nnn* Is the number of extend operations to be performed. The default number is 127.

Command	Description
SET HIPERFILE=*nnn*	*nnn* Is the maximum number of (4K) pages for an individual hiperspace size. The default is 524,288 (4K) pages or 2 GB in size. This parameter is equivalent to the SET FILELIM command. If both parameters are used and different limits are set, then the lower limit is enforced.
SET HIPERINSTALL={ON|OFF}	ON Installs HiperEDA. OFF Disables HiperEDA.
SET HIPERLOCKED={ON|OFF}	ON Disallows processing of user interface commands. OFF Allows processing of user interface commands (such as SET HIPEREDA). This value is the default.
SET HIPERSPACE=*nnn*	*nnn* Is the number of (4K) pages for aggregate hiperspace size. By default there is no limit. This parameter is equivalent to the SET TCBLIM command. If both parameters are used and different limits are set, then the lower limit is enforced.

Note: All of the SET options described (except for SET HIPEREDA) can only be executed from EDASPROF.

For more information on HiperEDA, see Chapter 5, *Configuring Server System Features*, in the *Server Configuration and Operations for MVS* manual.

CHAPTER 2

Server Security and Data Access Control

Topics:

- Connecting to a Server (Server Security)
- Security of Connected User
- Server Security Exits
- Data Access Control
- Attribute Summary

These topics describe server security and data access control. For more information about data access for specific data adapters, see the *Data Adapter Administration for MVS and VM* manual.

Connecting to a Server (Server Security)

A client attempting to connect to a server for either private or pooled deployment is subject to the security of the server. Before a connection can be made, the server must verify the identity of the client requesting the connection. The following are the connection processing types:

- **Already Verified Processing** relies on the server operating system to authenticate the user ID and password associated with the client request before attempting to connect to the server.
- **Explicit Verification Processing** requires that the client supply a user ID and encrypted password to the server for authentication.

Already Verified Processing

Already Verified Processing allows a server to authenticate the client application connection request based on user ID and node ID but not password information. The password is considered already verified if the operating system from which the request originates is a trusted node of the server.

Note: If you are using a FOCUS front-end, do not issue a REMOTE command to set the user ID and password.

Trusted nodes are operating systems considered to be secure by the server. Currently, the client operating systems that are identified to a server as trusted nodes are:

- MVS
- OS/390 and z/OS
- VM
- UNIX
- OpenVMS
- Windows NT/2000

Note: If the MVS Server is configured to allow this connection, see Chapter 8 of the *Server Configuration and Operations for MVS* manual for details on the TRUST_NT parameter.

Before you invoke Already Verified Processing, the following must be observed:

- The client application should not submit any user ID or password information within its connection request.
- The client's native login ID must also exist on the server platform and must appear in the same character case (upper, lower, or mixed).

The client API programmatically derives the client user ID from its login information, packages it with the client node ID, sets the Already Verified flag, and sends the connection request to the server.

When the server receives the connection request, it confirms that the Already Verified flag is set. It then checks if the client node ID is a trusted node, and that the client user ID exists on the server platform. If all these conditions are true, a connection is established. If any of these conditions are not true, the connection request is rejected, and an appropriate message is sent to the client.

Explicit Verification Processing

Explicit Verification Processing is invoked whenever a client application submits a connection request that contains user ID and password information. The password supplied as an argument is encrypted by the API, packaged with the user ID, and sent to the server.

The server receives the user ID and password, decrypts the password, then authenticates the user ID and password. The authentication is done through a call to the native security package. It is the responsibility of the server, and the operating system under which the server is running, to allow or reject the connection request.

If the user ID and password are authenticated, a connection is established between client and agent. If, however, the user ID or password is invalid or has expired, the connection request is rejected, and an appropriate message is sent to the client.

Security of Connected User

Depending on server deployment, the following security settings are used by the data access agent that processes the client's requests:

Deployment	Security Settings
Private | Isolated	The security settings of the connected user.

Pooled	You can control the security environment of all users who connect to a pooled server using the following optional parameters in the EDASERVE file:
	`USERID=userid`
	`PASSWORD=password`
	Note: If you do not use these parameters, the user ID of the person who submitted the MVS server job is used.

Server Security Exits

The server provides the following general-purpose security exits:

Type of Exit	Usage
External Database Logon Exit (XDBLGN)	Enables a site to install its own security package to be invoked in conjunction with operating system security upon a connection to the server.
Pre-Verify User ID Exit (PVUIDXT)	May be used prior to data access security authorization to customize a trusted node security scheme by matching input user IDs with the connecting partner nodes identified by a logical name and a network ID.
CSECT in FOCUS Exit (FOCUSID)	Called by the server to enforce a system-defined DBAPWD. Provides a server with a password that either can or cannot be overridden. This exit can perform a RACROUTE to obtain the DBAPWD value from a system security package.
External DB Security Exit (XDBSC)	The External DB Security Exit (XDBSC) may use the security group of a user requesting information from a database accessible through a server to restrict user access to defined portions of a given DB. This exit works in conjunction with the server's DBA security and replaces symbolic variables specified in VALUE restrictions with values supplied by the exit.
Security for VM Attach Manager (ATTCHMGR EDAREXX) **VM only.**	Enables a site to customize security so that the server authenticates the user ID and password.

External Database Logon Exit

The External Database Logon Exit (XDBLGN) enables a site to install its own security exit to be invoked with various operating system security packages. After the standard system logon (or verify-at-connect) procedure is performed, XDBLGN is invoked with the system user ID, password, and an optional new password, as well as a site-defined identifier (the external security group).

Based on a return code set by the exit, the server either disqualifies the user and terminates the connection or saves an exit-provided token in association with the new session and continues normal processing.

The server attempts to load the module XDBLGN during initialization processing. If the load is successful, the exit is called once for initialization, once per USER LOGON (or CONNECT), and once per USER LOGOFF (or DISCONNECT). The user-supplied logon exit must be re-entrant and will stay resident throughout the life of the server.

XDBLGN can determine the event to be processed by the function code (funcode) passed as its second parameter. Unused parameters are passed as null values. The use and applicability of the other parameters are dictated by the function code.

Reference

Parameters of Function Codes

Function Code	Description
`999`	Exit initialization. Instructs server for subsequent calling sequences.
`0, 1, or 2`	Performs user verification or disqualification. Possible values are: `0` is used for user connection passing system security authentication without password change. `1` is used for an authenticated user with a new password. `2` is used for a user that failed system security authentication. XDBLGN does not bypass external security authentication failure. In other words, XBDLGN will not connect a user who had failed external security authentication.
`-1`	Terminates user connection.

Syntax How to Call a Sequence for the XDBLGN Exit

```
XDBLGN( PVOID  exit_token,  /* Token set by exit during          */
                            /* initialization processing and     */
                            /* input for all subsequent calls.   */
                            /* Pointer to user data area         */
                            /* provided during the INIT EXIT     */
                            /* 8-byte area of user memory that   */
                            /* can be shared to allow a site     */
                            /* to coordinate services and can    */
                            /* be used as the repository of the  */
                            /* address of GETMAINed memory.      */
        PLONG  funcode,     /* =999 Exit Initialization          */
                            /* =0 NORMAL Connection,             */
                            /* =1 LOGON W/NEW PASSWORD           */
                            /* =2 Logon failed authentication    */
                            /* =-1 USER Disconnect               */
        char   userid[8],   /* A blank filled, upper cased       */
                            /* 8-byte character string           */
                            /* Not used for funcode=999          */
        char   secgrp[8],   /* A blank filled, upper cased       */
                            /* 8-byte character string           */
                            /* Not used for funcodes=999 and -1  */
        PCHAR  exiterrmsg,  /* Error Message Text, is displayed  */
                            /* in the EDAPRINT output. The       */
                            /* message must be null-terminated   */
                            /* and can be up to 81 characters    */
                            /* (including the null terminator).  */
                            /* Contains                          */
                            /* C-string(null term) <= 80 bytes   */
                            /* Not used for funcodes=999 and -1  */
        char   passwd[8],   /* A blank filled, upper cased       */
                            /* character string                  */
                            /* Not used for funcodes=999 and -1  */
        char   newpass[8],  /* A blank filled, upper cased       */
                            /* character string                  */
                            /* Not used for funcodes=999 and -1  */
        PPVOID  pToken,     /* A pointer to user security        */
                            /* handle which is GETMAINed by the  */
                            /* user's exit.                      */
                            /* Output by funcodes 0 and 2        */
                            /* Input to funcode  -1              */
                            /* Not used for funcodes=999         */
        PLONG   rc       ); /* A long pointer returned           */
                            /* by the exit                       */
                            /* rc=0 -> success, no message       */
                            /* rc=4 -> success, send msg text    */
                            /* rc=8 -> failure, send msg text    */
                            /* NOTE: msg text is sent only       */
                            /* with funcodes 0, 1, and 2         */
```

For MVS, a sample XDBLGN can be found in *qualif*.EDACTL.DATA(XDBLGNA).

Pre-Verify User ID Exit

The Pre-Verify User ID Exit (PVUIDXT) may be used to customize a trusted node security scheme. This exit matches a user ID with the connecting partner nodes identified by the following:

Communications Protocol	Partner Communications Identifier
LU6.2	The APPLID of the connecting session.
TCP/IP	The network ID (an alphanumeric representation of the IP address).

This exit allows or disallows connection to a server and may replace the input user ID with an ID appropriate for the host security subsystem. It may also provide an 8-byte password to be included in the verification procedure.

For example, a Windows client accesses MVS data through a UNIX Hub Server. With this configuration, the user ID and password can be verified on the UNIX Hub Server and a secure connection passed on to MVS. This configuration uses Already Verified Processing, providing trusted node security. This exit is invoked with all security modes set on a server.

The remainder of these topics provide an overview of how a configuration can be set up with a Hub Server installed as a trusted node. If this exit is not installed, no connections are rejected based on the partner communications ID.

Syntax How to Issue PVUIDXT Calling Sequences

In order for further validation of a user ID and a router address, the exit is called with the following calling sequence:

```
/*  pvuidxt( pPartner_Logical_Name, pPartner_Communications_ID, */
/*           pUserid, pPassword, pRC ) ;                        */
/*                                                              */
    void pvuidxt( char *, char *, char *, char *, long * ) ;
```

where:

pPartner_Logical_Name

Points to the string containing the connecting partner_entity_name.

pPartner_Communications_ID

Points to the string containing the communication ID for TCP/IP (the IP address). This is a variable length alphanumeric string. This value is not supplied for SNA connections.

pUserid

Points to an area containing the user ID provided by the client application. It is passed in uppercase. If the user exit replaces the incoming user ID, it must not use a length greater than or equal to the input string length.

pPassword

Points to an area, 8 bytes in length, into which the user exit can place a password associated with the incoming user ID from the specified partner node. If this password field contains a password to be used at the time of verification, pRC must be set equal to 1 (TRUSTED_EXIT_SUCCESS_USE_PSW).

pRC

Is the return code from the exit. Possible values are:

0 indicates the user ID/entity name combination has been validated and the user ID will be checked against your system security package.

1 indicates the user ID and password will be verified.

-1 indicates the validation check has failed and the user will be disconnected.

The trustos routine is called before PVUIDXT and determines whether the client operating system security is accepted by the server security system.

Syntax How to Define the Trustos Operating System

```
long trustos( unsigned char ) ;
```

where:

trustos

Returns a value of 0 or 1. A user who wants a different result must write a trustos routine.

0 indicates that the client operating system security is unacceptable.

1 indicates that the client operating system security is acceptable.

unsigned char

Is the already verified code for client applications.

0 indicates that the No already verified code is available.

1 indicates that the MVS client is already verified.

2 indicates that the OpenVMS client is already verified.

3 indicates that the UNIX client is already verified.

4 indicates that the VM client is already verified.

5 indicates that the Windows NT/2000 client is already verified.

Note: Your system administration staff can create their own conventions for other known security packages.

For MVS, a template for the PVUIDXT logon is located in *qualif*.EDACTL.DATA(PVUIDXTA).

CSECT in FOCUS Exit

The CSECT in FOCUS Exit (FOCUSID) provides the server with a DBA password that either can or cannot be overridden. For more information, see *Data Access Control* on page 2-19. It is called by the server to enforce a system-defined DBAPWD. This exit can perform a RACROUTE to obtain the DBAPWD value from a system security package.

Syntax How to Issue the CSECT in a FOCUS Exit Calling Sequence

```
CALL FOCUSID(reserved1,reserved2,DBAPWD,reserved3,RC)
```

where:

reserved1

Is reserved for future use.

reserved2

Is reserved for future use.

DBAPWD

Is the 8-character (8-byte) output (password) returned by the exit.

reserved3

Is reserved for future use.

RC

Is the return code set by FOCUSID.

0 indicates that the DBAPWD is posted by FOCUSID.

4 indicates that the external security system is not installed.

8 indicates that there is no DBAPWD field for this user.

16 indicates a fatal error in FOCUSID.

Note: DBAPWD is not set if RC is non-zero.

If the last non-blank character of the returned password is a period (.) character (x'4B'), then the password set is the string, without the period, and the password cannot be changed by a SET PASS or -PASS command. For example:

Password Returned by FOCUSID	Password Set As	User Restriction
SAMP1	SAMP1	Cannot be set by user.
SAMP2	SAMP2	Can be set by user.

Example Obtaining the Session User ID

The following sample exit uses the GETUSER general subroutine to obtain the session user ID. If it matches a static, hardcoded prefix, a DBAPWD is then set. An RC=0 is also set so the server will honor the DBAPWD. If the user ID does not match, the exit sets RC=8 to indicate that there is not a DBAPWD for this user. The sample FOCUSID user exit is located in *qualif*.EDACTL.DATA(FOCUSIDA).

```
FOCUSID  CSECT
            USING FOCUSID,R15
            B     SKIPHDR
            DC    CL8'FOCUSID'
            DC    CL8'&SYSDATE'
            DC    CL8'&SYSTIME'
SKIPHDR     DS    0H
*
            BAKR  R14,R0       SAVE REGS IN CALLER
*
            LR    R12,R15      SET BASE REF.
            USING FOCUSID,R12  IDENTIFY BASE REG
            DROP  R15
*
            LR    R2,R1        TEMPORARILY HOLD THE PARM REG.
            STORAGE OBTAIN,LENGTH=72  GET A SAVE AREA
            LR    R13,R1       POINT TO SAVE AREA
            LR    R1,R2        RESTORE PARM POINTER
            MVC   4(4,R13),FCID SAY WHO THIS IS IN THE SAVE AREA
*
            LR    R9,R1
            L     R3,8(,R9)    R3: DBAPWD
            L     R5,16(,R9)   R5: RC
*
*           The following code calls the supplied General Subroutine
*           GETUSER to obtain the current USERID for the session.
*           After obtaining the USERID it is compared to a static AUTHID
*           prefix for validation.
*
            CALL  GETUSER,GETUID
            CLC   AUTHUID,GETUID
            BE    SETDBA
*           Set RC = 8 to indicate no DBAPWD for this user
            MVC   0(4,R5),EIGHT    SET RD = 8
            B     RTRN
*
*           Logic to set a DBAPWD for the user
*
SETDBA    EQU   *
*
*          This code inserts a static DBAPWD for the user. You can
*          use RACROUTE, or another service, to obtain the DBAPWD from
```

```
*          your system security package.
*
           MVC   0(8,R3),DBAPWD   MOVE USERID TO 3RD ARG (DBAPWD)
           MVC   0(4,R5),ZERO     CET RC = 0
*
           Return logic
*
RTRN       EQU   *
           LR    R2,R13           SAVE AREA TO FREE
           STORAGE RELEASE,LENGTH=72,ADDR=(R2)  NOW FREE IT
*
           PR                     RESTORE STATE & RTRN TO CALLER
*
*
           SPACE

*--------------------------------------------------------*
*  DATA CONSTANTS                                        *
*--------------------------------------------------------*
            DS    0D
FCID        DC    C'FCID'      USED TO SAY WHO THIS IS
ZERO        DC    F'0'
FOUR        DC    F'4'
EIGHT       DC    F'8'
SIXTEEN     DC    F'16'
BLANK       DC    C'        '
DBAPWD      DC    C'TONY.   '
GETUID      DC    C'XXXXXXXX'
AUTHUID     DC    C'EDA'
*

R0          EQU   0
R1          EQU   1
R2          EQU   2
R3          EQU   3
R4          EQU   4
R5          EQU   5
R6          EQU   6
R7          EQU   7
R8          EQU   8
R9          EQU   9
R10         EQU   10
R11         EQU   11
R12         EQU   12
R13         EQU   13
R14         EQU   14
R15         EQU   15
*
            SPACE
            END
```

Password Exit Installation

The object code of FOCUSID, plus any other object code which it calls, is link-edited into load modules TABP with the JCL shown below. The new FOCUSID module replaces the dummy version contained in the distributed load modules.

Procedure How to Install the Password Exit

Follow these steps to install your password exit.

1. Write, compile, and link-edit your FOCUSID code.
2. Edit *qualif*.EDALIB.DATA(genefid), to your site's specifications, using the sample JCL shown below as a guide. Change the FOCUSID DD statement to point to your new FOCUSID module.
3. After editing *qualif*.EDALIB.DATA(genefid), submit the job, which creates the module FOCUSID and links the password interface modules into TABP.

Syntax How to Install the Password Exit

The following is a sample of *qualif*.EDALIB.DATA(genefid), which installs your interface.

```
//LINK1 EXEC   PGM=IEWL,PARM='LET,NCAL,LIST,MAP,SIZE=1024K'
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(10,1))
//OLDMOD    DD DSN=qualif.EDALIB.LOAD,DISP=SHR
//FOCUSID   DD DSN=user.compiled.focusid,DISP=SHR
//MAINTAIN  DD DSN=qualif.EDALIB.DATA,DISP=SHR
//SYSLMOD   DD DSN=qualif.TEST.EDALIB.LOAD,DISP=SHR
//SYSLIN    DD *
  MODE AMODE(31),RMODE(31)
  INCLUDE FOCUSID                     <---- FOCUSID CODE
  INCLUDE MAINTAIN(ACFINT1)           <---- MODE-SWITCHING CODE
  ENTRY ACFINT1                       <---- CNTL STATEMENT
  NAME FOCUSID(R)                     <---- NEW MODULE
        INCLUDE MAINTAIN(ACFINT0,ACFID)  <---- ACTIVATION PROGRAMS
        INCLUDE OLDMOD(TABP)             <---- MODULE TO BE CHANGED
        INCLUDE MAINTAIN(TABP)           <---- CNTL STATEMENTS
        NAME TABP(R)                     <---- NEW MODULE
```

where:

qualif

Is the high-level qualifier for the server production libraries.

user.compiled.focusid

Is the name of your new FOCUSID module.

External DB Security Exit

The External DB Security Exit (XDBSC) may use the security group of a user requesting information from a database accessible through a server to restrict user access to defined portions of a given DB. This exit works in conjunction with the server's DBA security and replaces symbolic variables specified in VALUE restrictions with values supplied by the exit. For more information on DBA security, see *Data Access Control* on page 2-19.

The External DB Security Exit supports two function codes:

- The first function code, Set DB Security Group, is called to determine the security group for a user to a specific database.
- The second function code, Translate Symbolic Value, is called for each symbolic variable specified in VALUE restriction expressions for the security group input.

Symbolic Variables are specified in the Master File and prefixed by an '@' character.

Example Specifying a Symbolic Value

```
FILE=SAMPLE      ,SUFFIX=SQLDS,$
SEGNAME=SAMPSEG,SEGTYPE=S0,$
   FIELD=USER_NO,UNO,A8,A8,$
END
DBA=TEST
USER=RALPH,ACCESS=R,RESTRICT=VALUE,NAME=SYSTEM,
      VALUE=USER_NO  EQ @C1,$
```

Example Using the External Database Security Exit (XDBSC)

```
XDBSC  (  PLONG Function Code,  /* Input to Exit.                    */
                                /* = 1 for Set DB Security Type      */
                                /* = 2 for Translate Symbolic Value  */
char Userid[8],                 /* Input to Exit.                    */
                                /* This should be authenticated      */
                                /* User ID provided at User Connect  */
char Secgrp [8],                /* A blank filled, upper-case        */
                                /* 8-byte character string.          */
                                /* For Set DB Security Group         */
                                /* function                          */
                                /* Input and Ouput to Exit.          */
                                /* On Input, Security Group that     */
                                /* user connected with or assigned   */
                                /* by the External DB Logon Exit     */
                                /* On output, Security Group that    */
                                /* should be used for subsequent     */
                                /* queries to the specified DB.      */
```

```
                                /* For Translate Symbolic bValue     */
                                /* function                          */
                                /* Input to Exit, usually based on   */
                                /* previous Set DB Security Group    */
                                  /* call.                            */

        PCHAR Table Name,       /* Input to Exit indicates the       */
                                /* table the user wishes to issue    */
                                /* queries against.                  */
                                /* Zero-delimited string.            */
                                /* Maximum length including 0= 20    */
        PCHAR Message,          /* Output from Exit. Error message   */
                                /* returned in case of failure.      */
                                /* Zero-delimited string.            */
                                /* Maximum length including 0=80     */
        PVOID Token,            /* Input to Exit.  This token is     */
                                /* the same as the one used in the   */
                                /* External DB Logon Exit.           */
                                /* Refers to user's security handle  */
                                /* from logon exit.                  */
        PLONG RC,               /* Output from Exit. Return Code     */
                                /* RC =  0   -> success              */
                                /* RC !=  0  -> failure, abort query */
                                /*                                   */
                                /* The following three parameters    */
                                /* are applicable only to            */
                                /* Translate Symbolic Value calls.   */
                                /* They should be passed as null     */
                                /* parameters Set DB Security Group  */
                                /* calls:                            */
        char Access [2]         /* Input to Exit, indicates users    */
                                /* access intention (R for Read only)*/
                                /* A blank filled, uppercased        */
                                /* character string.                 */
                                /*.The value supplied here will      */
                                /* replace the ACCESS = Value in     */
                                /* the DBA restriction in the MFD    */
        PCHAR SymName,          /* Input to Exit. Symbolic Name      */
                                /* without the '@' prefix.           */
                                /* Zero-delimited string.            */
                                /* Maximum length including 0= 68    */
        PCHAR SymValue          /* Output from Exit.  Values to be   */
                                /* substituted for Symbolic Name.    */
                                /* If multiple values are specified  */
                                /* they will be separated by 0. 00   */
                                /* will indicate end of list.        */
                                /* Maximum length including 0=4096   */
```

Security for the VM Attach Manager

If your site wishes to customize security so that the server authenticates the user ID and password, be sure to enable the security system. The security system is enabled by the presence of a file named ATTCHMGR EDAREXX, with a file mode A0, located on the Attach Manager's production disk. The ATTCHMGR EDAREXX file, if present, is passed control for processing the client user ID and password. Supplied on the installation tape and present on the maintenance disk is a sample version for VMSECURE, called ATTCHMGR SAMPVMSR. The ATTCHMGR SAMPVMSR file supplied on your distribution tape is shown below. Follow the instructions contained within the comments below to enable security.

Example Using the ATTCHMGR EDAREXX

```
/* This exec is created as a SAMPLE procedure for use with VMSECURE.   *
/* This exec will allow the ATTACH MANAGER to place the user on        *
/* VMSECURE HOLD after a number of invalid password calls.             *
/*                                                                     *
/* This exec is executed by the Server ATTACH MANAGER when security is *
/* enabled. The presence of this FILE, renamed to ATTCHMGR EDAREXX,    *
/* on the Server ATTACH MANAGER, enables the VM SECURITY exit.         *
/*                                                                     *
/* The number of times a CONNECT may present a USERID and PASSWORD     *
/* to the ATTACH MANAGER for VALIDATION before the userid is placed    *
/* on HOLD is defined as 5. This may be changed in EDANAMES CONFIG     *
/* in the LOGONATTEMPT field.                                          *
/*                                                                     *
/* The SERVER ATTACH MANAGER requires VMSECURE DIAGPCHK or             *
/* CLASS B CP PRIVILEGES.                                              *
/* The ATTACH MANAGER also requires MAINT privileges for the list of   *
/* AGENTS                                                              *
/* which the ATTACH MANAGER is to AUTOLOG / MANAGE.                    *
/*                                                                     *
/* The following are SAMPLE RULES which are defined in VMSECURE CONFIG *
/* and must be modified with the correct names defined by each site    *
/* for the ATTACH MANAGER and SERVERS. This MUST be done by the SITE   *
/* SECURITY OFFICER.                                                   *
/*                                                                     *
/* 1. The name EDALIST can be any name which conforms to the site      *
/* naming convention.                                                  *
/* 2. The list of SERVERS, which FOLLOW the EDALIST entry, is the same *
/* as the AUTOLOG entries in the EDANAMES CONFIG file.                 *
/* 3. EDAMGRX is the name of the SERVER ATTACH MANAGER                 *
/*                                                                     *
/* LIST *EDALIST EDASERV1 EDASERV2 EDASERV3 EDASERV4 EDASERV5          *
/* GRANT MAINTMAN OVER *EDALIST HOLD TO EDAMGRX                        *
/* GRANT DIAGPCHK TO EDAMGRX                                           *
/* GRANT NOPASS TO EDAMGRX                                             *
/*                                                                     *
```

```
Address 'COMMAND'
Trace 'O'

Arg user passwrd.

failcnt = 5      /* number of times a user can issue a bad password    *
/* set a GLOBALV VARIABLE so we read the EDANAMES CONFIG file only     *
/* once per IPL session. If LOGONATTEMPT entry is missing, then We     *
/* default to the 'failcnt' value of 5. If found then this becomes the *
/* new default value.                                                  *

'GLOBALV SELECT ATTCHMGR GET CNFGCNT' /* get count from CONFIG         *
If cnfgcnt = ' '
Then Do
'FINIS EDANAMES CONFIG *' /* close the config file                     *
/* read config file                                                    *

'EXECIO * DISKR EDANAMES CONFIG * ( FINIS STEM EDACFG.'
Do ix = 1 To edacfg.0 By 1
If edacfg.ix = ' '
Then Iterate /* skip blank lines                                       *
comment = Index(edacfg.ix,';')
If comment = 1
Then Iterate /* whole line is comment                                  *
If comment = 0
Then data = edacfg.ix /* no ; in data line                            *
Else data = Substr(edacfg.ix,1,comment) /* data B4 ; in line           *
Upper edacfg.ix
Parse Value edacfg.ix With keyword '=' count .
If keyword = 'LOGONATTEMPT' & Datatype(count) = 'NUM'
Then Do
failcnt = count
Leave ix
End
End ix
'GLOBALV SELECT ATTCHMGR SET CNFGCNT' failcnt /* valid for IPL         *
End
Else Do
failcnt = cnfgcnt /* set 'failcnt' to GLOBALV value                    *
End
/* currently DIAGA0 will return 3 return codes.                        *
/* 0 - password and userid valid                                       *
/* 4 - userid valid, but password is not correct                       *
/* 8 - userid is not valid. userid is probably on HOLD.                *
rcdiag = DIAGA0(user,passwrd) /* valid password and userid             *
Select
When rcdiag = 0 /* if user supplies correct password clear old entries *
Then Do
```

```
'GLOBALV SELECT' user 'PURGE'
exitrc = 0 /* let the user thru                                          *
End

When rcdiag = 4 /* invalid password                                      *
Then Do
'GLOBALV SELECT' user 'GET COUNT' /* get current count                  *
If count = ' '
Then count = '0'
currcnt = count + 1 /* add 1 to current "try" count                     *
If currcnt >= failcnt /* too many tries - HOLD the user                 *
Then Do
'VMSECURE MAINT MANAGE' user 'HOLD' /* place id on HOLD                 *
If rc /= 0
Then Do
Say "VMSECURE COMMAND MAINT MANAGE" user "HOLD" ,
"failed RC=" rc
End
Else Do
'GLOBALV SELECT' user 'PURGE' /* allow LOGON after                    *
End /* HOLD is removed                                                 *
End
Else Do /* dont lock out yet - update cnt                              *
'GLOBALV SELECT' user 'SETL COUNT' currcnt
End
exitrc = 4 /* fail logon                                               *
End
When rcdiag = 8 /* userid invalid                                       *
Then Do
exitrc = 8 /* fail logon                                               *
End
Otherwise
Do
Say "Unexpected RC from DIAGA0 RC=" diagrc
exitrc = 8 /* fail logon                                               *
End
End /* select */
Exit exitrc
```

The default behavior for the supplied VMSECURE exit is to put a user ID on HOLD if there are five (5) consecutive invalid logon attempts for that ID. The LOGONATTEMPT parameter, located in the EDANAMES CONFIG file, is used to modify the number of invalid logon attempts allowed before the user ID is locked. See the EDANAMES CONFIG section later on in this chapter for further details on the parameter.

Note: The ATTCHMGR EDAREXX file must only be made available to the Attach Manager. Having the ATTCHMGR EDAREXX file available to the server IDs can cause unpredictable results.

Reference EDANAMES CONFIG

The EDANAMES CONFIG file identifies a list of server IDs to the ATM. The EDANAMES:

- PROTOCOL = TCP
- CFGFILE = EDACSG ; DDNAME in the EDATCP EXEC

The SERVERS line has to precede the AUTOLOG list. The number of servers can be larger than the number of IDs to be autologged.

`SERVERS=nn`

Is the maximum number of concurrent servers allowed.

```
AUTOLOG id1
AUTOLOG id2
AUTOLOG id3
AUTOLOG id4
AUTOLOG id5
```

`LOGONATTEMPT=nn`

Is the maximum number of times a logon may be attempted before SERVER is placed on HOLD.

`WAKEUP=nn`

The server will do host maintenance every nn minutes.

`PREAUTOLOG=YES`

PREAUTOLOG servers in the server list.

`EXEC EDASERV`

Is the exec to run on the server when autologged.

The EDANAMES CONFIG file contains no default value for the LOGONATTEMPT parameter. There is, however, a default of 5 attempts coded in the supplied VMSECURE security exit, ATTCHMGR SAMPVMSR. To override the ATTCHMGR coded default, uncomment the LOGONATTEMPT parameter line and specify the desired value.

No External Security Manager

If your site does not have an external security manager, you must assign class B privileges to the ATM and activate the ATTCHMGR SAMPNONE security exit. This provides the ATM the ability to verify the logon password against native VM security. The ATTCHMGR security exit is activated by changing the File Type to EDAREXX.

Server User Exits

The server provides the following user exits to use with the VSAM Data Adapter:

- Dynamic and Re-Entrant Private User Exit (GETPRV).
- Re-entrant VSAM Compression Exit (ZCOMP).

Each of these exits is described in the VSAM Data Adapter topics.

Data Access Control

The following topics apply to a Hub Server and a Full-Function Server. They do not apply to a server configured as with the SET SQLENGINE = Command.

File Access Control

In any computer system, it is important that data be secured from unauthorized access. Master Files provide access control mechanisms to prevent misuse and corruption of data by ensuring that users have access to only those resources for which they have explicit authorization.

Server access control complements and extends security rules that may already exist in the (R)DBMS, and respects restrictions previously defined by an underlying security system, such as RACF, CA-ACF2, CA-TOP SECRET, OpenVMS Security, or UNIX Security. These packages typically work at the file or table level.

Implementing server access control features is a process in which you specify:

- The names or passwords of server users who have been granted access to a file.
- The type of access the user is granted.
- The segments, fields, or ranges of data values to which the user's access is restricted.

You provide server access control on a file-by-file basis.

Note: If the Master File contains security statements, the SQL translation feature, Automatic Passthru will be disabled. Therefore, it is recommended not to use this feature for relational data sources

Data Source Access Control

The server provides a variety of access control options in addition to existing operating system and data source security:

Reference **Access Control Options**

Using the…	You can…
DBA attribute	Assign the name or password of the Database Administrator for the file. The Database Administrator has unlimited access to the file.
USER attribute	Limit the users who have access to a given file. Only users whose name or password you specify in the Master File with access control placed on it has access to that file.
ACCESS attribute	Define a user's access rights to read, write, or update. • RW, allows a user to read from and write to a file. • R, allows a user to only read data in a file. • W, allows a user to only write new instances to a file. • U, allows a user to only update records in a file.
RESTRICT attribute	• Restrict a user's access to certain fields or segments. • Ensure that only records that pass a validation test are retrieved. • Limit the values a user can alter or write to the data source.
DBAFILE attribute	Point to passwords and restrictions stored in another Master File.

Example Describing Access Control Options

```
FILENAME=PERS, SUFFIX=FOC                                          ,$

SEGMENT=IDSEG,                  SEGTYPE=S1                         ,$
 FIELD=SSN         ,ALIAS=SSN    ,FORMAT=A9                         ,$
 FIELD=FULLNAME    ,ALIAS=FNAME  ,FORMAT=A40                        ,$
 FIELD=DIVISION    ,ALIAS=DIV    ,FORMAT=A8                         ,$

SEGMENT=COMPSEG, PARENT=IDSEG, SEGTYPE=S1,$
 FIELD=SALARY      ,ALIAS=SAL    ,FORMAT=D8                         ,$
 FIELD=DATE        ,ALIAS=DATE   ,FORMAT=YMD                        ,$
 FIELD=INCREASE    ,ALIAS=INC    ,FORMAT=D6                         ,$
END

DBA=JONES76                                                        ,$

USER=TOM           ,ACCESS=RW                                      ,$
USER=BILL          ,ACCESS=R    ,RESTRICT=SEGMENT  ,NAME=COMPSEG   ,$
USER=JOHN          ,ACCESS=R    ,RESTRICT=FIELD    ,NAME=SALARY    ,$
                  NAME=INCREASE                                    ,$
USER=LARRY         ,ACCESS=U    ,RESTRICT=FIELD    ,NAME=SALARY    ,$
USER=TONY          ,ACCESS=R    ,RESTRICT=VALUE    ,NAME=IDSEG,
   VALUE=DIVISION EQ 'WEST'                                        ,$
```

The declarations (called access control declarations), including and following the word END, inform the server that you need security for the file and what type of security you want.

You describe your file security by specifying various values for these access control attributes in a comma-delimited format, just as you specify any other attribute in the Master File.

The word END, on a line by itself in the Master File, terminates the segment and field attributes and indicates that the access limits follow. If you place the word END in a Master File, you must follow it by at least a DBA attribute.

Identifying the DBA: The DBA Attribute

The first line of access limits must be a name that identifies the Database Administrator.

Syntax How to Identify the DBA

```
DBA=dbaname,$
```

where:

dbaname

Is an arbitrary code name for the user. It can contain up to a maximum of eight characters and is terminated by the delimiter (,$).

Example Specifying a DBA Attribute

```
DBA=JONES76,$
```

The Database Administrator (DBA) has the freedom to change any of the access control attributes. If you change the DBA password in the Master File, you must use the RESTRICT command in a stored procedure to inform the server that this change has been made. Unless this is done, the server will assume this is an attempt to bypass the restriction rules.

Note:

- Every file with access limits must have a DBA.
- The Database Administrator has unlimited access to the file and all cross-referenced files. Therefore, you cannot specify field, segment, or value restrictions with the DBA attribute.

You cannot encrypt and decrypt Master Files or restrict existing data files without setting the correct DBA password in the procedure using the encryption commands. You can have different DBA passwords for each data source.

Example Using DBA in a Stored Procedure

```
-DEFAULT &1=OLDDBA, &2=FILENAME, &3=NEWDBA
SET PASS=&1
RESTRICT
&2
END
SET PASS=&3
```

Note: You should validate every access control attribute before you use it in a file. It is particularly important to verify that the VALUE limits are correct. Value tests execute as if they were extra screening conditions typed after each request statement. Since users are unaware of the value limits, errors caused by the value limits may confuse them.

Identifying Users: The USER Attribute

The USER attribute is an arbitrary code name that identifies users who have legitimate access to the file. You cannot specify a USER attribute alone. You must follow it by at least an ACCESS restriction to specify what sort of ACCESS to grant to the user.

Before using a secured file, a user must enter a password using the SET PASS or SET USER command in a stored procedure or through an API command. If you do not include the password in the Master File, the user is denied access to the file. When the user does not have a password or has one that is inadequate for the type of access requested, the following message displays:

```
(EDA047) THE USER DOES NOT HAVE SUFFICIENT ACCESS RIGHTS TO THE FILE:
filename
```

Any user whose name or password is not declared in the Master File is denied access to that file.

Syntax How to Identify the User

```
USER=username,
```

where:

username

Is an arbitrary code name. It can consist of a maximum of eight characters. You can specify a blank password. A blank password does not require the user to issue a SET PASS= command. A user with a blank password may still have access limits and this is convenient when a number of users have the same access rights.

Example Identifying a User

```
USER=TOM,...
```

An example of setting a user's password to blank, and access to read only follows:

```
USER= , ACCESS=R,$
```

Establishing User Identity

A user must enter a password before using any Master File that has access control specified for it. A single user may have different passwords in different files. For example, in file ONE the rights of password BILL apply, but in file TWO the rights of password LARRY apply. Use the SET PASS command to establish the passwords.

Syntax How to Establish User Identity

```
SET {USER|PASS}=name [[IN{file|*[NOCLEAR]}], name [IN file]...]
```

where:

name

Is the user's name or password.

file

Is the file name that the password applies to.

*

Indicates that name replaces all passwords active in all files.

NOCLEAR

Provides a way to replace all passwords in the list of active passwords while retaining the list.

Example Establishing User Identity # 1

In the following example, the password TOM is in effect for all files that do not have a specific password designated:

```
SET PASS=TOM
```

Example Establishing User Identity # 2

For the next example, in file ONE the password is BILL, and in file TWO the password is LARRY. No other files have passwords set:

```
SET PASS=BILL IN ONE, LARRY IN TWO
```

Here, all files have password SALLY except files SIX and SEVEN, which have password DAVE:

```
SET PASS=SALLY, DAVE IN SIX
SET PASS=DAVE IN SEVEN
```

The password is MARY in file FIVE and FRANK in all other files:

```
SET PASS=MARY IN FIVE,FRANK
```

The server maintains a list of the files for which a user has set specific passwords. To see the list of files, issue the following in a stored procedure:

```
? PASS
```

When the user sets a password IN * (all files), the list of active passwords collapses to one entry with no associated file name. To retain the file name list, use the NOCLEAR option.

Example Establishing User Identity # 3

In the next example, the password KEN replaces all passwords active in all files, and the table of active passwords is folded to one entry:

```
SET PASS=KEN IN *
```

In the following, MARY replaces all passwords in the existing table of active passwords (which consists of files NINE and TEN) but FRANK is the password for all other files. The option NOCLEAR provides a shorthand way to replace all passwords in a specific list:

```
SET PASS=BILL IN NINE,TOM IN TEN
SET PASS=MARY IN * NOCLEAR,FRANK
```

Users must issue their passwords using the SET PASS or SET USER command in a stored procedure for each server session in which they use a secured file. They may issue their passwords at any time before using the file and can issue a different password afterward to access another file.

Specifying the Type of Access: The ACCESS Attribute

The ACCESS attribute specifies what sort of access to the file a user is granted. Every access control declaration, except the DBA, must have a USER and ACCESS attribute.

Access levels affect what kind of commands a user can issue. Before you decide what access levels to assign to a user, you must consider what commands that user will need. If a user does not have sufficient access rights to use a given command, the following message displays:

```
(EDA047) THE USER DOES NOT HAVE SUFFICIENT ACCESS RIGHTS TO THE FILE:
filename
```

ACCESS levels determine what a user can do to the file. You use the RESTRICT attribute to limit access to fields, values, or segments. Every USER attribute must be assigned an ACCESS attribute. The RESTRICT attribute is optional. Without it, the user has unlimited access to fields and segments in the file with the appropriate authority.

Example Specifying the Access Type

```
USER=TOM, ACCESS=RW,$
```

where:

`TOM`

Is the user's name or password.

`ACCESS=R`

Read only

`ACCESS=W`

Write only.

`ACCESS=RW`

Read the file and write new instances.

`ACCESS=U`

Update only

DECRYPT and ENCRYPT Commands

Only users with the DBA password can issue a DECRYPT or ENCRYPT command.

Limiting the Access: The RESTRICT Attribute

The ACCESS attribute tells a server what a user can do with a file. The optional RESTRICT attribute further restricts a user's access to certain fields, values, or segments.

Syntax How to Limit Access With RESTRICT

```
RESTRICT={restrict}, NAME={name|SYSTEM} [,VALUE=test] ,$
```

where:

restrict

Restricts a user's access to certain fields, values, or segments. Possible values are:

FIELD specifies that the fields named with the NAME attribute cannot be accessed by the user.

SEGMENT specifies that the segments named with the NAME attribute cannot be accessed by the user.

SAME specifies that the user has the same restrictions as the user named in the NAME attribute. No more than four nested SAME users are valid.

NOPRINT specifies that the field named in the NAME or SEGMENT attribute can be mentioned in a request statement, but does not display.

VALUE specifies that the user can have access to only those values that meet the test described in the *test* attribute.

name

Is the name of the field or segment you wish to restrict. When used after NOPRINT, this can only be a field name.

NAME=SYSTEM

Can be used only with value tests and restricts every segment in the file, including descendant segments. You can specify multiple fields or segments by issuing the RESTRICT attribute several times for one user.

test

Is the value test that the data must meet before the user can have access.

Example Restricting a User's Access

```
USER=BILL ,ACCESS=R ,RESTRICT=SEGMENT ,NAME=COMPSEG,$
```

Restricting Fields and Segments

The RESTRICT attribute identifies the segments or fields that the user cannot access. Anything not named in the RESTRICT attribute is accessible.

Without the RESTRICT attribute, the user has access to the entire file. Users may be limited to reading, writing, or updating new records, but every record in the file is available for the operation.

Syntax How to Restrict Access to a Field or Segment

```
RESTRICT=restrict, NAME=name,$
```

where:

restrict

Restricts a user's access to certain fields, values, or segments. Possible values are:

FIELD specifies that the fields named with the NAME attribute cannot be accessed by the user.

SEGMENT specifies that the segments named with the NAME attribute cannot be accessed by the user.

SAME specifies that the user has the same restrictions as the user named in the NAME attribute.

NOPRINT specifies that the field named in the NAME or SEGMENT attribute can be mentioned in a request statement, but does not display. When used after NOPRINT, NAME can only be a field name.

name

Is the name of the field or segment you want to restrict. When used after NOPRINT, this can only be a field name.

Example Restricting Access to a Segment

In the following example, Bill has read-only access to everything in the file except the COMPSEG segment:

```
USER=BILL ,ACCESS=R ,RESTRICT=SEGMENT ,NAME=COMPSEG,$
```

Note:

- If a field or segment is mentioned in the NAME attribute, it cannot be retrieved by the user. If such a field or segment is mentioned in a request statement, it is rejected as beyond the user's access rights. With NOPRINT, the field or segment can be mentioned, but the data is not displayed. The data appears as blanks for alphanumeric format or zeros for numeric fields.
- You can restrict multiple fields or segments by providing multiple RESTRICT statements. For example, if you wish to restrict Harry from using both field A and segment B, you issue the following access limits:

  ```
  USER=HARRY, ACCESS=R,  RESTRICT=FIELD  , NAME=A,$
                         RESTRICT=SEGMENT, NAME=B,$
  ```

- You can restrict as many segments and fields as you like.
- Using RESTRICT=SAME is a convenient way to reuse a common set of restrictions for more than one password. If you specify RESTRICT=SAME and provide a user name or password as it is specified in the USER attribute for the NAME value, the new user will be subject to the same restrictions as the one named in the NAME attribute. You can then add additional restrictions as they are needed. In the following example, both Sally and Harry have the same access privileges as BILL. In addition, Sally is not allowed to read the SALARY field.

  ```
  USER=BILL  ,ACCESS=R ,RESTRICT=VALUE ,NAME=IDSEG ,
       VALUE=DIVISION EQ 'WEST'                    ,$

  USER=SALLY ,ACCESS=R ,RESTRICT=SAME  ,NAME=BILL  ,$
                        RESTRICT=FIELD ,NAME=SALARY,$

  USER=HARRY ,ACCESS=R ,RESTRICT=SAME  ,NAME=BILL  ,$
  ```

Restricting Values

You can restrict the values to which a user has access by providing a test condition in your RESTRICT statement. The user is restricted to using only those values that satisfy the test condition.

You can restrict values in one of two ways: you can restrict the values the user can read from the file, or you can restrict what the user can write to a file. These restrictions are two separate functions: one does not imply the other. You use the ACCESS attribute to specify whether the values the user reads or the values the user writes are restricted.

You restrict the values a user can read by setting ACCESS=R and RESTRICT=VALUE. This type of restriction prevents the user from seeing any data values other than those that meet the test condition provided in the RESTRICT statement. A RESTRICT statement with ACCESS=R functions as an involuntary WHERE statement in a SELECT. Therefore, the syntax for ACCESS=R value restrictions must follow the rules for a WHERE test in a report request.

Syntax How to Restrict the Values a User Can Read

```
ACCESS=R, RESTRICT=VALUE, NAME=name, VALUE=test,$
```

where:

name

Is the name of the segment on which you are performing the tests.

test

Is the test being performed.

Example Restricting Values a User Can Read

With the following restriction, Tony can only see records from the western division:

```
USER=TONY ,ACCESS=R ,RESTRICT=VALUE ,NAME=IDSEG,
     VALUE=DIVISION EQ 'WEST' ,$
```

You type the test expression after VALUE=. The syntax of the test condition is the same as that used by the SQL SELECT predicate, except the word WHERE does not precede the phrase and the operator EQ replaces the equal sign (=). Should several fields have tests performed on them, separate VALUE statements must be provided. Each test must name the segment to which it is applied. For example:

```
USER=RAY ,ACCESS=R ,RESTRICT=VALUE ,NAME=IDSEG,
     VALUE=DIVISION EQ 'EAST' or 'WEST',$
                                  NAME=IDSEG,
     VALUE=SALARY LE 10000,$
```

If a single test condition exceeds the allowed length of a line, it can be provided in sections. Each section must start with the attribute VALUE= and end with the terminator (,$). For example:

```
USER=SAM, ACCESS=R, RESTRICT=VALUE ,NAME=IDSEG,
     VALUE=DIVISION EQ 'EAST' OR 'WEST',$
     VALUE=OR 'NORTH' OR 'SOUTH',$
```

Note: The second and subsequent lines of a value restriction must begin with the keyword OR.

You can apply the test conditions to the parent segments of the data segments on which the tests are applicable. Consider the following example:

```
USER=RAY ,ACCESS=R ,RESTRICT=VALUE ,NAME=IDSEG,
     VALUE=DIVISION EQ 'EAST' OR 'WEST',$
                             NAME=IDSEG,
     VALUE=SALARY LE 10000, $
```

The field named SALARY is actually part of a segment named COMPSEG. Since the test specifies NAME=IDSEG, the test is effective for requests on its parent, IDSEG. In this case, the request to print FULLNAME would only print the full names of people who meet this test, that is, whose salary is less than or equal to $10,000, even though the test is performed on a field that is part of a descendant segment of IDSEG. If, however, the test was made effective on COMPSEG, that is, NAME=COMPSEG, then the full name of everyone in the file could be retrieved, but with the salary information of only those meeting the test condition.

Syntax How to Restrict Values a User Can Write

```
...ACCESS=W, RESTRICT=VALUE, NAME=name, VALUE=test,$
```

where:

name

Is an arbitrary value.

test

Is the test being performed.

```
USER=CHUCK ,ACCESS=W ,RESTRICT=VALUE ,NAME=CHRANGE,
     VALUE=SALARY LT 20000 AND SALARY GT 5000,$
...ACCESS=W, RESTRICT=VALUE, NAME=name, VALUE=test,$
```

name

Is the name of the segment on which you perform the test.

test

Is the test being performed.

```
USER=CHUCK ,ACCESS=W ,RESTRICT=VALUE ,NAME=COMPSEG,
     VALUE=SALARY LT 20000 AND SALARY GT 5000,$
...ACCESS=W, RESTRICT=VALUE, NAME=name, VALUE=test,$
```

name

Is the name of the segment on which you perform the test.

test

Is the test being performed.

```
USER=CHUCK ,ACCESS=U ,RESTRICT=VALUE ,NAME=IDSEG,
     VALUE=D.DIVISION EQ 'EAST' ,$
VALUE=DIVISION EQ 'EAST'
```

Example Restricting Both Read and Write Values

```
USER=TILLY ,ACCESS=R ,RESTRICT=VALUE ,NAME=IDSEG,
     VALUE=DIVISION EQ 'NORTH',$
            ACCESS=W ,RESTRICT=VALUE ,NAME=DIVTEST,
     VALUE=DIVISION EQ 'NORTH',$
```

Placing Access Control Information in a Central File: The DBAFILE Attribute

The DBAFILE attribute enables you to place all of the passwords and restrictions for many Master Files in one central file. Each individual Master File points to this central control file. Groups of Master Files with the same DBA password may share a common DBAFILE, which has the same DBA password. There are several benefits to this technique. The primary ones are:

- Passwords only have to be stored once when they are applicable to a group of data files. This simplifies password administration.
- Files with different DBA passwords can now be joined. In addition, individual DBA information remains in effect for each file in a join.

The central DBAFILE is a standard Master File. Other Master Files can use the password and access control restrictions listed in the central file by specifying its file name with the DBAFILE attribute.

Syntax How to Place Access Control Information in a Central File

```
END
DBA=dbaname, DBAFILE=dbafile ,$
```

where:

dbaname

Is the same as the dbaname in the central file.

dbafile

Is the name of the central file.

Example Placing Access Control Information in a Central File

You can specify passwords and restrictions in a DBAFILE that apply to every Master File that points to that DBAFILE. You can also include passwords and restrictions for specific Master Files by including FILENAME attributes in the DBAFILE. The following example shows a group of Master Files that share a common DBAFILE named FOUR:

```
ONE MASTER
FILENAME=ONE
END
DBA=ABC, DBAFILE=FOUR,$

TWO MASTER
FILENAME=TWO
END
DBA=ABC, DBAFILE=FOUR,$

THREE MASTER
FILENAME=THREE
END
DBA=ABC,
DBAFILE=FOUR,$

FOUR MASTER
FILENAME=FOUR,$
SEGNAME=mmmmm,$
FIELDNAME=fffff,$
END
DBA=ABC,$
   PASS=BILL,ACCESS=R,$
   PASS=JOE,ACCESS=R,$
FILENAME=TWO,$
   PASS=HARRY,ACCESS=RW,$
FILENAME=THREE,$
   PASS=JOE,ACCESS=R,RESTRICT=...,$
   PASS=TOM,ACCESS=R,$
```

Note:

- All Master Files that specify the same DBAFILE have the same DBA password.
- The central DBAFILE is a standard Master File. It may include additional attributes before the END statement that signifies the presence of DBA information. The DBA password in the DBAFILE is the same as the password in all the Master Files that refer to it. This prevents individuals from substituting their own access control. All of these Master Files should be encrypted.
- The DBAFILE may contain a list of passwords and restrictions following the DBA password. These passwords apply to all files that reference this DBAFILE. In the example above, PASS=BILL, with ACCESS=R (read only), applies to all files that contain the attribute DBAFILE=FOUR.
- After the common passwords, the DBAFILE may specify file-specific passwords and additions to general passwords. You implement this feature by including FILENAME attributes in the DBA section of the DBAFILE (for example, FILENAME=TWO).
- File-specific restrictions override general restrictions for the specified file. In the case of a conflict, passwords in the FILENAME section take precedence. For example, a DBAFILE might contain ACCESS=RW in the common section, but specify ACCESS=R for the same password by including a FILENAME section for a particular file.
- Value restrictions accumulate; all value restrictions must be satisfied before retrieval. In the preceding example, note the two occurrences of PASS=JOE. JOE is a common password for all files, but in FILENAME=THREE it carries an extra restriction, RESTRICT=…, which applies only to file THREE.

CREATE/DROP SYNONYM does not maintain DBA information in the file.

Reference DBAFILE File Naming Requirements

When a DBAFILE includes a FILENAME attribute for a specific Master File, the FILENAME attribute in the referencing Master File must be the same as the FILENAME attribute in the DBA section of the DBAFILE. This prevents users from renaming a Master File to a name not known by the DBAFILE. For example:

```
ONE MASTER
FILENAME=XONE
END
DBA=ABC, DBAFILE=FOUR,$

FOUR MASTER
FILENAME=FOUR
END
DBA=ABC,$
FILENAME=XONE,$
```

ONE MASTER is referred to in requests as SELECT ... FROM ONE. However, both ONE MASTER and the DBA section of the DBAFILE, FOUR MASTER, specify FILENAME=XONE.

Reference Connection to an Existing DBA System

If there is no mention of the new attribute, DBAFILE, there will be no change in the characteristics of an existing system. In the current system, when a series of files is joined, the first file in the list is the controlling file. Its passwords are the only ones examined. All files must have the same DBA password.

In the new system, the DBA sections of all files in a join structure are examined. If DBAFILE is included in a Master File, then its passwords and restrictions are read. To make the DBA section of a file active in a join list, specify DBAFILE for that file.

Once you start to use the new system, you should convert all of your Master Files. For Database Administrators who want to convert existing systems but do not want a separate physical DBAFILE, the DBAFILE attribute can specify the file itself. For example:

```
FILENAME=SEVEN,
  SEGNAME=...
    FIELDNAME=...
     .
     .
     .
END
DBA=ABC,DBAFILE=SEVEN,$    (OR DBAFILE= ,$)
 PASS=...
 PASS=...
```

Reference Combining Applications

Since each file now contributes its own restrictions, you can join files that come from different applications and have different DBA passwords. The only requirement is a valid password for each file. You can therefore grant access rights for one application to the control of a different DBA by assigning a password in your system.

Reference Summary of Access Control Attributes

The following is a list of all the access control attributes used by a server:

Attribute	Alias	Maximum Length	Meaning
DBA	DBA	8	Value assigned is the code name of the Database Administrator (DBA) who has unrestricted access to the file.
USER	PASS	8	Values are arbitrary code names, identifying users for whom access control restrictions will be enforced.
ACCESS	ACCESS	8	Levels of access for this user. Values are: R read only W write new segments only RW read and write U update values only
RESTRICT	RESTRICT	8	Types of restrictions to be imposed for this access level. Values are: SEGMENT FIELD VALUE SAME NOPRINT
NAME	NAME	48	Name of segment or field restricted.
VALUE	VALUE	80	Test expression that must be true when RESTRICT=VALUE is the type of limit.
DBAFILE	DBAFILE	8	Names the Master File that contains passwords and restrictions to use.

Encrypting Files

Since the restriction information for a data source is stored in the Master File, you will want to encrypt the Master File in order to prevent users from examining the restriction rules. Only the Database Administrator can encrypt a Master File. Additional files that might need encrypting are Dialogue Manager procedures, global profiles, group profiles, and user profiles.

To encrypt a file, use the ENCRYPT command in a stored procedure. The process can be reversed if you wish to change the restrictions. The command to restore the Master File to a readable form is DECRYPT. The DBA password must be issued with the SET command before the file can be encrypted or decrypted.

Syntax How to Encrypt and Decrypt Files

```
{ENCRYPT|DECRYPT} FILE filename type
```

where:

`ENCRYPT`

Encrypts the file.

`DECRYPT`

Decrypts the file.

`filename`

Is the name of the file, without its extension, to be encrypted or decrypted.

`type`

Indicates the type of file to be encrypted/decrypted. Possible values are:

`EDAPROF` indicates a profile with the extension of .prf.

`MASTER` indicates a synonym with the extension of .mas.

`FOCEXEC` indicates a Dialogue Manager procedure with the extension of .fex.

Example Encrypting the edasprof.prf File

```
SET PASS=SECRET
ENCRYPT FILE edasprof EDAPROF
```

Stored Procedure Access Control

Most data security issues are best handled by the server access control facility. Nevertheless, some additional data security facilities are incorporated in Dialogue Manager. These are:

- Setting passwords in encrypted stored procedures.
- Defining variable passwords.
- Encrypting and decrypting stored procedures.
- Locking stored procedure users out of the server.

Syntax How to Set Passwords in Encrypted Stored Procedures

```
-PASS password
```

You do not need to issue the password with the SET command. It also means that the password is not visible to anyone. The procedure must be encrypted so that printing the procedure cannot reveal the password. For more information, see the *Stored Procedure Reference* manual.

Syntax How to Define Variable Passwords

The Dialogue Manager command -PASS can have a variable attached to it as well as a literal.

```
-PASS &value
```

For example:

```
-PASS &MYPASS
```

This command is only visible when you edit the stored procedure. It does not appear when the ECHO option is ALL and is not printed in a batch run log. Note that ECHOed data is returned as messages to the client.

Encrypting and Decrypting Stored Procedures

You may want to keep the actual text of a stored procedure confidential while allowing users to execute the stored procedure because there is confidential information within the stored procedure, or because you do not want the stored procedure changed by unauthorized users. You can protect a stored procedure from unauthorized users with the ENCRYPT command. Any user can execute an encrypted stored procedure, but you must decrypt the stored procedure to view it. Only a user with the DBA password can decrypt the stored procedure.

You use the following to encrypt the stored procedure named SALERPT:

```
SET PASS=DOHIDE
ENCRYPT FILE SALERPT FOCEXEC
```

The stored procedure can only be viewed by decrypting it, as follows:

```
SET PASS=DOHIDE
DECRYPT FILE SALERPT FOCEXEC
```

Note: You must issue these commands in a stored procedure.

Attribute Summary

The following pages describe the access control attributes.

Syntax How to Use Access Control Attributes

```
END
DBA=dbaname [,DBAFILE=dbafile],$
USER=username1, ACCESS=access, RESTRICT=restrict, NAME=name
[,VALUE=test],$
USER=username2, ...
```

Attribute:	ACCESS
Alias:	ACCESS
Length:	8 characters.
Permitted Values:	R, W, RW, U
Example:	`ACCESS=R`
Function:	Defines the level of access the user is permitted. Possible values are: `R` allows the user to only read the file. `W` allows the user to only write to the file. `RW` allows the user to read from and write to the file. `U` allows the user to only update existing values. If no RESTRICT attribute is specified, the ACCESS attribute will give the user access to the entire file. If you wish to further restrict the user's access to certain fields, values, or segments, you must use both the ACCESS attribute and the RESTRICT attribute.
Changes:	Changes cannot be made to the access control attributes without the DBA password.

Attribute:	DBA
Alias:	DBA
Length:	8 characters.
Permitted Values:	Any alphanumeric character.
Example:	`DBA=JONES76`
Function:	The value assigned is the code name of the Database Administrator (DBA), who has unrestricted access to the file.
Changes:	Changes cannot be made to the access control attributes without the DBA password.

Attribute:	DBAFILE
Alias:	DBAFILE
Length:	8 characters.
Permitted Values:	Any Master File name. All files referencing the same DBAFILE must have identical DBA passwords.
Example:	`DBAFILE=FILE1`
Function:	The value identifies the Master File that contains passwords and restrictions.
Changes:	Changes cannot be made to the access control attributes without the DBA password.

Attribute:	NAME
Alias:	NAME
Length:	48 characters.
Permitted Values:	Any segment name or field name in the file.
Example:	`NAME=IDSEG`
Function:	The value assigned is the name of the segment or field from which the user is restricted. If RESTRICT=SAME, NAME must be the name of another user in the file.
Changes:	Changes cannot be made to the access control attributes without the DBA password.

Attribute: RESTRICT

Alias: RESTRICT

Length: 8 characters.

Permitted Values: `SEGMENT, FIELD, VALUE, SAME, NOPRINT`

Example: `RESTRICT=VALUE`

Function: The value assigned specifies the type of restriction to be imposed for a given user and access level. The meaning of each value follows:

`FIELD`

Specifies that the fields named in the NAME attribute cannot be accessed by the user.

`SEGMENT`

Specifies that the segments named in the NAME attribute cannot be accessed by the user.

`SAME`

Specifies that the user has the same restrictions as the user named in the NAME attribute.

`NOPRINT`

Specifies that the field named in the NAME or SEGMENT attribute can be mentioned in a request statement but will not be displayed.

`VALUE`

Specifies that the user can have access to only those values that meet the test described in the VALUE attribute. This restriction can either restrict the user from reading certain values or restrict the user from writing or altering certain values.

Changes: Changes cannot be made to the access control attributes without the DBA password.

Attribute:	USER
Alias:	PASS
Length:	8 characters.
Permitted Values:	Any alphanumeric character, including blanks
Example:	`USER=FRED12`
Function:	The value assigned is an arbitrary code name that identifies the user.
Changes:	Changes cannot be made to the access control attributes without the DBA password.

Attribute:	VALUE
Alias:	VALUE
Length:	80 characters.
Permitted Values:	Any value permitted in a VALIDATE or IF … condition, including GT, LT, EQ, NE, LE, GE, IF, as well as field names and numbers.
Example:	`VALUE=DIV NE ' ' AND DATE GT 0`
Function:	This is a test expression that must be true when RESTRICT=VALUE is the type of limit. Records that do not meet the test condition will not be retrieved.
Changes:	Changes cannot be made to the access control attributes without the DBA password.

CHAPTER 3

Metadata Services With SQLENGINE SET

Topics:

- How Applications Access Metadata
- Maintaining Upward Compatibility

When the server is dedicated to accessing one Relational Database Management System (RDBMS) using the SET SQLENGINE command in the global profile, metadata calls to the server are processed against the native catalogs of the RDBMS.

How Applications Access Metadata

The metadata procedures used by applications to query the native catalog directly are:

- For an ODBC-enabled application, ODBC SQL calls.
- For an API-enabled application, EDARPC for ODBC*xxxx* calls.

The following table shows the relationship between the ODBC call and the API call

ODBC Call	API Call
SQLTables	ODBCTABL
SQLColumns	ODBCCOLS
SQLPrimaryKeys	ODBCPKEY
SQLStatistics	ODBCSTAT
SQLProcedures	ODBCPROC
SQLProcedureColumns	ODBCPRCC
SQLSpecialColumns	ODBCSCOL
SQLColumnPrivileges	ODBCCPRV
SQLForeignKeys	ODBCFKEY
SQLTablePrivileges	ODBCTPRV

You can use the following two commands to control or override table and column metadata calls:

```
ENGINE sqlengine SET ODBCCOLSSORT
ENGINE sqlengine SET ODBCTABL
```

You can include these commands in any of the supported server profiles.

Obtaining Column Information (DB2 only)

When you issue either the SQLColumns or ODBCCOLS Metadata call from a client application, by default, the server requests the columns from DB2 in *colno* order.

Syntax How to Request the Sort Order of Column Data

```
ENGINE DB2 SET ODBCCOLSSORT {ON|OFF}
```

where:

ON

The column data is requested from the server with order by column. This value is the default.

OFF

The column data is returned in unsorted order.

Obtaining a User-Defined Metadata

When a client application issues either an ODBC or API metadata call, the server runs internal procedures that issue default SQL against the native RDBMS catalogs. You can issue your own SQL to run in place of the default SQL. You can specify this type of override for any of the internal server routines that deal with metadata.

Syntax How to Request User-Defined Metadata

```
ENGINE sqlengine SET ODBCxxxx procname
```

where:

sqlengine

Indicates the data source. You can omit this value if you previously issued the SET SQLENGINE command.

ODBC*xxxx*

Is the internal server routine name. Possible values are: ODBCTABL, ODBCCOLS, ODBCPKEY, ODBCSTAT, ODBCPROC, ODBCPRCC, ODBCSCOL, ODBCCPRV, ODBCFKEY, and ODBCTPRV.

procname

Is the procedure to run when the server receives the metadata call. This procedure must be available through the FOCEXEC ddname allocation in the server JCL.

Note: If this override procedure is used on a server running under MVS, it will add approximately 600K of storage above the line for each user.

When coding an override procedure, care must be taken to maintain the select list (answer set layout). The select list must conform to the ODBC specification for the return from the SQLTables call. See the *ODBC 2.0 Programmer's Reference and SDK Guide* for the SQLTables specification layout. Also, when using this override procedure, the parameters that are sent with the Metadata call need to be parsed correctly.

The override procedure should take the following format:

1. Dialogue Manager code to parse metadata call parameters.
2.
```
SQL sqlengine
SELECT code;
```
3.
```
TABLE
ON TABLE PCHOLD FORMAT ALPHA
END
```

Items 1 and 2 above are user coded; item 3 must always be present at the end of the procedure as coded above.

Example Returning a List of Tables

The following sample code found in *qualif*.EDARPC.DATA(DB2ODBC1) returns a list of tables that the connected user is authorized to INSERT, UPDATE, DELETE, and SELECT. It is a sample of how to code a DB2 override procedure for ODBCTABL. It is one of several ways to code SQL to return an answer set for the ODBCTABL call. You can code any relevant query as long as the SELECT list is maintained.

In a server profile, issue ENGINE DB2 SET ODBCTABL DB2ODBC1, and then execute the following RPC request on the server:

```
ODBCTABL  ,<NULL>,,,,0,0,*
```

The following example matches the above ODBCTABL call:

```
-*
-* Dialogue Manager code to parse the ODBCTABL parameter list
-*
-DEFAULTS 1=' ',2='%',3='%'
-IF &2 NE '<NULL>' THEN GOTO LAB1;
-SET &2 = '%';
-LAB1
-IF &3 NE ' ' THEN GOTO LAB2;
-SET &3 = '%';
-LAB2
-*
-* SQL SELECT code
-*
SQL DB2
SELECT ' ',T2.CREATOR,T2.NAME,'TABLE',' '
FROM SYSIBM.SYSTABAUTH T1,SYSIBM.SYSTABLES T2
WHERE T1.GRANTEE  = USER
AND   T1.TTNAME   LIKE '&2'
AND   T1.TCREATOR LIKE '&3'
AND   T2.TYPE     = 'T'
AND  (T1.DELETEAUTH IN  ('G','Y')
OR    T1.INSERTAUTH IN  ('G','Y')
OR    T1.SELECTAUTH IN  ('G','Y')
OR    T1.UPDATEAUTH IN  ('G','Y'))
AND   T1.TTNAME   = T2.NAME
AND   T1.TCREATOR = T2.CREATOR
UNION
SELECT ' ',T2.CREATOR,T2.NAME,'VIEW',' '
FROM SYSIBM.SYSTABAUTH T1,SYSIBM.SYSTABLES T2
WHERE T1.GRANTEE  = USER
AND   T1.TTNAME   LIKE '&1'
AND   T1.TCREATOR LIKE '&2'
AND   T2.TYPE     = 'V'
AND  (T1.DELETEAUTH IN  ('G','Y')
OR    T1.INSERTAUTH IN  ('G','Y')
OR    T1.SELECTAUTH IN  ('G','Y')
OR    T1.UPDATEAUTH IN  ('G','Y'))
AND   T1.TTNAME   = T2.NAME
AND   T1.TCREATOR = T2.CREATOR
ORDER BY CREATOR,NAME;
-*
-* The following code must always be present
-*
TABLE
ON TABLE PCHOLD FORMAT ALPHA
END
```

Maintaining Upward Compatibility

In Version 4.3.x or earlier, an Extended Catalog (SYSOWNER table) was created at installation time. This provided additional control to the list of tables returned with the SQLTables or ODBCTABL call. In Version 5.1.0 and higher, this table is no longer created. If you need to use this table from a previous version of the server to continue to have control over the list of tables, issue the following in any supported server profile:

```
ENGINE sqlengine SET OWNERID ownerid
```

where:

sqlengine

Indicates the data source. You can omit this value if you previously issued the SET SQLENGINE command.

ownerid

Identifies the creator or owner of the Extended Catalog table SYSOWNER. If this command is used, the SQL generated to provide a list of tables will be limited by the owner names in the SYSOWNER table.

This command is only supported for customers who have configured a Relational Gateway in previous releases of the server and want to continue with the same configuration.

CHAPTER 4

Metadata Services in Full-Function and Hub Servers

Topics:

- Creating Master and Access Files (Full-Function and Hub Servers)
- Dynamic Catalog Overview
- Maintaining the Dynamic Catalog
- Dynamic Catalog Tables

The server catalog provides real-time descriptions of objects used by applications. Just as a Relational Database Management System (RDBMS) has a native catalog, a server has its own catalog for keeping track of data sources, stored procedures, and other objects.

The data in a catalog is called metadata, because it is data that describes data available to the user.

Metadata Services on a Full-Function or Hub Server uses the server Dynamic Catalog to provide metadata to users. The server Dynamic Catalog provides data about data sources, collections of data sources, stored procedures, and other objects. The Dynamic Catalog is created when an instance of a Hub or Full-Function Server is configured.

Creating Master and Access Files (Full-Function and Hub Servers)

In order to access a data source on a Full-Function or Hub Server, a Master and Access File must exist for the data source:

- The Master File describes the columns of the data source, for example, column names and formats.
- The Access File includes additional parameters that complete the definition of the data source, for example, the full name and location.

Master and Access Files can represent an entire table or part of a table. Several pairs of Master and Access Files can define different subsets of columns for the same table. For non-relational data sources, one pair of Master and Access Files can describe several tables. In this manual, the term table refers to both base tables and views in data sources.

You can create Master and Access Files using any of the following facilities:

- **Catalog Administrator.** Catalog Administrator (available for relational data sources or SUFFIX=EDA data sources from a Hub Server to a Sub server only) is an interactive client/server application used to maintain and report from the Dynamic Catalogs. It issues the CREATE SYNONYM command to automatically create Master and Access Files including descriptive information and dynamically create an alias reference (synonym) to the data source.
- **Auto Tools.** The server Auto Tools use existing definitions of tables and views from the native non-relational data source to automatically generate Master and Access Files. The catalog definitions provide the column information, such as column names, data types, column lengths, and whether null values are allowed. Auto Tools exist for the following data sources:
  - AUTOADBS (Adabas)
  - AUTODBC (CA-Datacom/DB)
  - AUTOIDMS (CA-IDMS)
  - AUTONMD (Nomad)
  - AUTOIMS

  Note: Use the EDAAUTO procedure to access the Auto Tools.
- **Cobol FD Translator.** The Cobol FD Translator automatically creates Master Files from Cobol File Descriptions (FDs). In some cases, you may need to edit the Master File attributes that were created by the translator. See the data adapter topics for information on the attributes and parameters required for the specific data source. For more information on the Cobol FD Translator, see the *Cobol FD Translator* manual.
- **Manual editing.** You can manually create Master and Access Files using any system editor.

What Is a Synonym?

Synonyms allow an Enterprise to define unique names (or aliases) for each table that is accessible from a Hub or Full-Function Server. They are useful because they hide location information and the identity of the underlying data source from client applications.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master and Access File.

Creating Synonyms for Data Sources

The Catalog Administrator allows you to create synonyms for existing local or remote data sources and automatically adds this metadata to the Dynamic Catalog.

- Local data sources. Relational data sources on the server to which you are connected.
- Remote data sources. Resides on a subserver.

When the Catalog Administrator creates the synonym for the data source, it creates a Master File and an Access File for the data source. The Master File contains information about the columns in the data source, and the Access File contains information used to locate the data source.

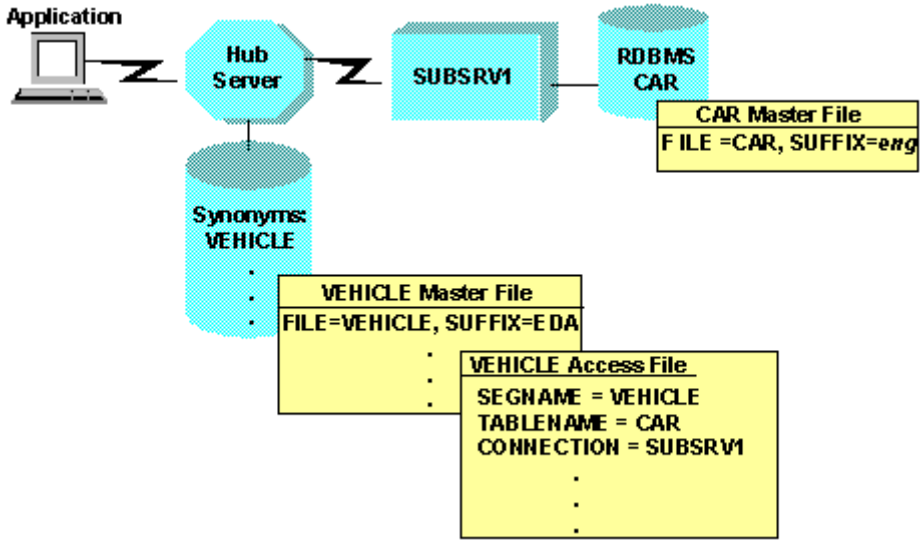
- If the data source is local, the Master File contains the attribute SUFFIX=*eng*, where *eng* indicates the relational database in which the data resides.
- If the data source is remote, the Master File contains the attribute SUFFIX=EDA. See the specific data adapter topics for details on the format and contents of Master Files and Access Files.

The following diagram illustrates the use of Master Files and Access Files for data sources. The Data Administrator creates the synonym VEHICLE on the Hub Server for the remote data source CAR, which resides on a Full-Function Server. The process automatically creates a VEHICLE Master File and VEHICLE Access File on the Hub Server.

- The VEHICLE Master File contains SUFFIX=EDA to indicate that the data source is remote.
- The VEHICLE Access File includes the synonym for the data source, the name of the data source, and the location of the data source:

```
SEGNAME = VEHICLE
TABLENAME = CAR
CONNECTION = SUBSRV1
```

When you connect to the Hub Server, the VEHICLE Master File and Access File are used to access the remote data source CAR, located on SUBSRV1.



In the above Access File example, the value SUBSRV1 of the CONNECTION parameter must match an outbound node block in the Hub Server's communications file. For more information, see the *Server Configuration* manual for your platform.

Providing Descriptive Information

By default, the CREATE SYNONYM command writes Master Files with the REMARKS, DESCRIPTION, TITLE, and HELPMESSAGE optional attributes.

This feature extends the metadata that is available in the server catalog by adding a table descriptor (REMARKS), and database creator (DBMS_CREATOR) to SYSTABLE, and column descriptors (REMARKS, TITLE, and HELPMESSAGE), and database creator (DBMS_CREATOR) to SYSCOLUM. Currently, the attributes will be included in a Master File when issuing:

- CREATE SYNONYM locally for Oracle, DB2, IDMS/SQL, and Teradata.

  For example, when creating a synonym locally for Oracle and DB2, if a table description exists the synonym created will contain a REMARKS attribute in the Master File. If a column description exists, the column in the synonym will contain a DESCRIPTION attribute.

- CREATE SYNONYM for remote subservers.

  For remote synonym creation, the attributes are included only if the Master File on the subserver contains the attributes. You can manually add the attributes to the synonym as well.

Syntax How to Control New Master File Attributes

```
SET SYNONYM={EXTENDED|BASIC}
```

where:

`EXTENDED`

Includes the attributes in the Master File when issuing a CREATE SYNONYM. This is the default value. It is not recommended that you change this setting without first consulting with your local representative.

`BASIC`

Does not include the attributes in the Master File when issuing a CREATE SYNONYM.

Note: A setting of BASIC changes the way the server obtains the metadata for a local relational table. If you use BASIC, access authority to the relational table that the synonym is being created for is required for the connected user.

The CREATE SYNONYM Command

If you do not use the Catalog Administrator to create (and/or delete) your synonyms, the CREATE SYNONYM (and/or DROP SYNONYM) command can be executed using a stored procedure on either the Hub or Full-Function Server.

Syntax How to Configure Remote Access to a Data Source

When the server is configured for access to a data source on a subserver, you can create a synonym for the data source using the following syntax:

```
CREATE SYNONYM appname/synonym FOR datasource AT server[NOCOLS][WITHSTATS]
END
```

where:

appname

Is a 1-8 character name. If the server is APP enabled, then an application must be provided.

synonym

Is a 1-64 character alias for the data source.

datasource

Is the fully-qualified name for the physical data structure.

server

Is the name of the subserver by which the synonym accesses the data source (maximum eight characters).

`NOCOLS`

Optionally specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`WITHSTATS`

Using this option, the number of rows in the target table of the CREATE SYNONYM command is recorded in the access file as CARDINALITY= nnn where nnn is number of rows. This value, if it exists is the Access File is used to optimize join processing.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: The CREATE SYNONYM command can span more than one line. However, a single element cannot span more than one line.

CREATE SYNONYM for Direct Access to a Data Source

When the Hub or Full-Function Server is configured for direct access to a data source, you can create a synonym for the data source in one of two ways:

- Including the DBMS server name in the synonym.
- Omitting it from the synonym and relying on DBMS-specific environment settings.

Syntax How to Configure Direct Access to a Data Source

```
CREATE SYNONYM [appname/] synonym FOR datasource DBMS eng [AT
dbms_server][NOCOLS][WITHSTATS]
END
```

where:

`datasource`

Is the fully-qualified name for the physical data structure, such as *location.ownerid.tablename* in DB2.

`eng`

Is the name of the data source's database management system (DBMS) installed on the server machine.

`AT dbms_server`

Is the name used by the DBMS for the database server on which the data source resides. When the server creates the synonym, this command becomes the value for CONNECTION= in the Access File. Use this option when dynamically switching server access from one DBMS server to another.

If, after creating the synonyms with this option, the DBMS server's environment changes (for example, if tables are moved to a different DBMS server), you must either create new synonyms or edit existing ones.

If this parameter is omitted, the server uses DBMS-specific settings in its environment.

`NOCOLS`

Optionally specifies that the Master File created for the synonym should not contain column information. If this option is used, the column data is retrieved dynamically from the data source at run time of the SQL request.

`END`

Indicates the end of the command, and is required on a separate line in the stored procedure.

Note: CREATE SYNONYM can span more than one line. However, a single element cannot span more than one line.

The DROP SYNONYM Command

The DROP SYNONYM command deletes a synonym from a Hub Server. Synonyms cannot be overwritten. You must first drop a synonym before using CREATE SYNONYM to create a revised Master File and Access File.

Syntax How to Drop a Synonym

```
DROP SYNONYM [appname/] synonym
```

where:

appname

Is a 1-8 character name. If the server is APP enabled, then an application must be provided.

synonym

Is the synonym to drop.

Location of Synonyms

The location of synonym Files is dependent on the use of Application Namespaces.

If the server is not APP enabled, the Master File is placed in a library (partitioned data set) allocated to the ddname EDASYNM. The Access File is placed in a library allocated to the ddname EDASYNA. If these ddnames are added at the JCL level, they remain in effect for all users of the server. If they are allocated dynamically with a profile or in a SERVINIT, you can customize the allocations for each user or group of users.

If the server is APP enabled, the Master File is placed in the relevant Application Namespace MASTER.DATA using the appname supplied at CREATE SYNONYM time. The Access File is placed in the relevant Application Namespace ACCESS.DATA.

Generating Server Profiles for Full-Function and Hub Servers

One of the functions performed by the installation procedures EDACFGH and EDACFGF is to create the following members in qualif.EDAPROF.DATA:

- EDACFGH

  (Howner_id)

  (Mowner_id)

  (Aowner_id)

- EDACFGF

  (Fowner_id)

  (Mowner_id)

  (Aowner_id)

These default profiles should be used under the conditions described in Mowner_id, Howner_id, Fowner_id, and Aowner_id Profiles.

- **Mowner_id Profile.** The Mowner_id profile is for the use of the server DBA whose responsibility it is to use Catalog Administrator to create synonyms for the non-App enabled server community. This profile has the following dynamic allocations, EDASYNM that points to *qualif*.owner_id.EDASYNM.DATA, and EDASYNA that points to *qualif*.owner_id.EDASYNA.DATA.

  By default, the EDASYNM and EDASYNA DDnames are also allocated in the Howner_id and Fowner_id profiles. Synonyms can now be created and used immediately on any default service.

  Once the synonym has been tested, it can be copied to *qualif*.EDAMFD.DATA and *qualif*.EDAAFD.DATA, which are allocated by default in the server JCL to ddnames MASTER and ACCESS, respectively.

- **Howner_id and Fowner_id.** The Howner_id (Hub Server) and Fowner_id (Full-Function Server) profiles are used for user connections to the server.

  To allow for both users and the DBA to access the same server, two services are set up by the configuration routines, which allocate the different profiles created at configuration time.

The following are examples of the two services, one for user connections and another for DBA use. This example is for a Hub Server. For a Full-Function Server, change the Howner_id to Fowner_id.

```
*****************************************************************
*                  Generic Service Block                        *
*****************************************************************
SERVICE=EDAUSER
PROGRAM=TSCOM3
NUMBER_READY = 0
DEPLOYMENT   = PRIVATE
REFRESH_LIMIT= 100
MAXIMUM=nnn
TRUST_NT=N
SERVINIT=*,++
   DYNAM ALLOC FILE EDASPROF -
        DA qualif.EDAPROF.DATA(Hownerid) SHR REU
  DYNAM ALLOC FILE EDAPROF  -
        DA qualif.EDAPROF.DATA          SHR REU
*****************************************************************
**        The following 2 allocations are for tracing          **
*****************************************************************
* DYNAM ALLOC FILE IBITRACE -
*       DA qualif.INSTALL.DATA(IBITRACE) SHR REU
++
*****************************************************************
*                  EDADBA Service Block                         *
*****************************************************************
SERVICE=EDADBA
PROGRAM=TSCOM3
NUMBER_READY = 0
DEPLOYMENT   = PRIVATE
REFRESH_LIMIT= 100
MAXIMUM=1
SERVINIT=*,++
   DYNAM ALLOC FILE EDASPROF -
        DA qualif.EDAPROF.DATA(Mownerid) SHR REU
```

```
*****************************************************************
**        The following 2 allocations are for tracing          *
*****************************************************************
* DYNAM ALLOC FILE IBITRACE -
*       DA qualif.EDACTL.DATA(IBITRACE) SHR REU
++
```

- **Aowner_id.** The Aowner_id profile is used for both DBA and user connections to the server in APP enabled mode.

The following is an example of a block used for an APP enabled service.

```
*****************************************************************
*                  EDAAPPS Service Block                        *
*****************************************************************
SERVICE      = EDAAPPS
PROGRAM      = TSCOM3
NUMBER_READY = 0
DEPLOYMENT   = PRIVATE
REFRESH_LIMIT= 100
MAXIMUM      = 16
TRUST_NT     = N
SERVINIT     =*,++
   APPROOT   = approotvalue
   DYNAM ALLOC FILE EDASPROF -
        DA qualif.EDAPROF.DATA(Aownerid) SHR REU
   DYNAM ALLOC FILE EDAPROF  -
        DA qualif.EDAPROF.DATA          SHR REU
*****************************************************************
**        The following allocation is for tracing              **
*****************************************************************
* DYNAM ALLOC FILE IBITRACE -
*       DA qualif.INSTALL.DATA(IBITRACE) SHR REU
++
```

See the *Server Configuration and Operations for MVS* manual for further details regarding the configuration of a Hub or Full-Function Server.

Enhanced Metadata Reporting

The Dynamic Catalog interface provides enhanced access to details about the metadata objects available on the server. The Dynamic Catalog table SYSFILES can be used to report on metadata and server objects. It can be used to replace the FOCMAP utility, which is no longer available. A SYSFILES reporting object can be any logical name in the form of a DDname. The following command can be used to set the object type:

```
ENGINE FMI SET SYSFILES object
```

where:

object

Is the DDname on which to report.

Syntax How to Retrieve Information From the Dynamic Catalog Table SYSFILES

Issue the SQL FMI SET SYSFILES command in any of the supported server profiles.

- To retrieve information on all Master Files and synonyms in the server's path, issue:

  ```
  ENGINE FMI SET SYSFILES MASTER
  ```

- To retrieve information on Master Files pointed to by EDASYNM, issue:

  ```
  ENGINE FMI SET SYSFILES EDASYNM
  ```

- To retrieve information on stored procedures, issue:

  ```
  ENGINE FMI SET SYSFILES FOCEXEC
  ```

  Use the following method as a replacement for the FOCMAP utility:

  ```
  DYNAM ALLOC FILE SYSFILES DA pds_name SHR REU
  ENGINE FMI SET SYSFILES SYSFILES
  ```

  Then issue the following SQL statement to retrieve a list of members from the PDS:

  ```
  SELECT FILENAME FROM SYSFILES
  ```

  If the PDS contains data members rather than load modules, ISPF statistics can also be obtained using SYSFILES. Refer to *SYSFILES (Full-Function and Hub Servers)* on page 4-20.

Dynamic Catalog Overview

The Dynamic Catalog consists of the following tables:

- *Dynamic* tables that are generated using a server internal metadata interface. The metadata interface generates the metadata whenever a client application queries the dynamic tables. Therefore, the most recent metadata is always available and is reflected in the response to the client's query.

Table	Objects Described
SYSTABLE	Data sources in the enterprise.
SYSCOLUM	Columns in data sources.
SYSKEYS	Columns that uniquely identify rows in data sources (key columns).
SYSINDEX	Indexed columns in data sources.
SYSFILES	Information on Master Files or stored procedures.

These dynamic tables have a corresponding Master File, which contains the attribute SUFFIX=FMI. This attribute indicates that the metadata interface retrieves information to satisfy the client request.

- *Static* tables that are implemented as server internal data sources. These tables must be maintained in order to reflect additions and deletions to the run-time environment. These tables are maintained with the Catalog Administrator.

Table	Objects Described
SYSRPC	Stored procedures available for execution and their parameters.
SYSCOLLN	Collections of data sources.
SYSCOLLT	Data sources in a collection.

The configuration procedure creates these static tables in the Dynamic Catalog.

Procedure How Applications Access Metadata

The metadata procedures used by applications to query the native catalog directly are:

- For an ODBC-enabled application, ODBC SQL calls.
- For an API-enabled application, EDARPC for ODBC calls.

The following table shows the relationship between the ODBC call and the API call:

ODBC Call	API Call
SQLTables	ODBCTABL
SQLColumns	ODBCCOLS
SQLPrimaryKeys	ODBCPKEY
SQLStatistics	ODBCSTAT
SQLProcedures	ODBCPROC
SQLProcedureColumns	ODBCPRCC
SQLSpecialColumns	ODBCSCOL
SQLColumnPrivileges	ODBCCPRV
SQLForeignKeys	ODBCFKEY
SQLTablePrivileges	ODBCTPRV

Maintaining the Dynamic Catalog

The Catalog Administrator is an interactive client/server application used to maintain and report from the Dynamic Catalog on either a Hub Server or Full-Function Server.

The Catalog Administrator enables a data administrator to:

- Create, maintain, and drop synonyms for data sources. The data source can be either remote or local.
- Create, maintain, and drop collections. A collection is a logical view of multiple physical databases and tables. Collections can be used by front-end applications in order to simplify the way that data is accessed.
- Maintain catalog information about stored procedures and their parameters, in order to make them accessible from a server.
- Generate reports containing details of the metadata for data sources, collections, and stored procedures.

The Catalog Administrator is a single-user application. Generally, only the data administrator should maintain a catalog. The server provides exclusive use of the temporary work files required during a Catalog Administrator session, thus preventing simultaneous catalog updates by multiple users. The Catalog Administrator is thus an administrator's tool, not a user-level tool.

Maintaining Collections (Full-Function and Hub Servers)

A collection is a group of related data sources (for example, a restricted number of tables) that a client application is allowed to access. Some connector products use this feature to limit the scope of data access. For a connector to use this feature, the collection must be created on the server to which the connector will be connected.

You can create collections on a Hub or Full-Function Server using the Catalog Administrator.

Maintaining Stored Procedures (Full-Function and Hub Servers)

A stored procedure resides on a server and can be executed by a client application. It can be written in Dialogue Manager or in a programming language such as C or COBOL. It can also be supplied by a connector.

Stored procedure metadata must be maintained on every server from which the stored procedure is invoked. Detailed information about a stored procedure and its parameters is entered once on the server on which the stored procedure resides.

See the *Stored Procedures Reference* manual for more information on user-written stored procedures.

Dynamic Catalog Tables

The Dynamic Catalog consists of the following dynamic tables (SUFFIX=FMI):

- SYSTABLE
- SYSCOLUM
- SYSKEYS
- SYSINDEX
- SYSFILES

The Dynamic Catalog consists of the following static tables:

- SYSRPC
- SYSCOLLN
- SYSCOLLT

The contents of each table are described in the following topics.

SYSTABLE (Full-Function and Hub Servers)

SYSTABLE contains one row for each data source identified to a server.

Column Name	Data Type	Description
NAME	CHAR(64	The synonym or data source.
CREATOR	CHAR(8)	Is always EDADBA.
TYPE	CHAR(1)	Reserved for system use.
DBNAME	CHAR(8)	Reserved for system use.
TSNAME	CHAR(8)	Reserved for system use.
DBID	INT	Reserved for system use.
OBID	INT	Reserved for system use.
COLCOUNT	INT	Reserved for system use.
EDPROC	CHAR(8)	Reserved for system use.
VALPROC	CHAR(8)	Reserved for system use.
CLUSTERTYPE	CHAR(1)	Reserved for system use.

Column Name	Data Type	Description
CLUSTERRID	INT	Reserved for system use.
CARD	INT	Reserved for system use.
NPAGES	INT	Reserved for system use.
PCTPAGES	INT	Reserved for system use.
IBMREQD	CHAR(1)	Reserved for system use.
REMARKS	CHAR(78)	The table description.
PARENTS	INT	Reserved for system use.
CHILDREN	INT	Reserved for system use.
KEYCOLUMNS	INT	Reserved for system use.
RECLENGTH	INT	Reserved for system use.
STATUS	CHAR(1)	Reserved for system use.
KEYOBID	INT	Reserved for system use.
LABEL	CHAR(30)	Reserved for system use.
CHECKFLAG	CHAR(1)	Reserved for system use.
CHECKRID	CHAR(4)	Reserved for system use.
AUDITING	CHAR(1)	Reserved for system use.
CREATEDBY	CHAR(8)	Reserved for system use.
LOCATION	CHAR(18)	Reserved for system use.
TBCREATOR	CHAR(8)	Reserved for system use.
TBNAME	CHAR(18)	Reserved for system use.
CATDATE	YYMMDD	Reserved for system use.
CATTIME	CHAR(6)	Reserved for system use.
TBTYPE	CHAR(8)	Reserved for system use.
PDSNAME	CHAR(50)	Reserved for system use.
SELFREFS	INT	Reserved for system use.

Column Name	Data Type	Description
DBMS_CREATOR	CHAR(8)	The real database owner of the table.
SERVER	CHAR(64)	The server= name from the acx file.
REALNAME	CHAR(64)	The real table name from the acx file.
ENCRYPTED	CHAR(1)	File encrypted? "Y" : "N".

SYSCOLUM (Full-Function and Hub Servers)

SYSCOLUM contains one row for each column in every data source identified to the server.

Column Name	Data Type	Description
TBNAME	CHAR(64)	The synonym or data source that contains the column.
TBCREATOR	CHAR(8)	Authorization ID of the owner of the synonym or data source containing the column.
DBMS_CREATOR	CHAR(8)	The real database owner of the table.
NAME	CHAR(66)	Name of the column.
COLNO	INT	The sequence number of the column in the metadata.
COLTYPE	CHAR(8)	Type of column: CHAR, DECIMAL, INTEGER, FLOAT, DOUBLE, DATE, BLOB, CLOB. DECIMAL is used for both packed and zoned types.
LENGTH	INT	The length of the column.
SCALE	INT	The maximum number of digits to the right of the decimal.
NULLS	CHAR(1)	Enter Y or N to indicate whether the column can contain null data.
COLCARD	INT	Reserved for system use.
HIGH2KEY	CHAR(8)	Reserved for system use.
LOW2KEY	CHAR(8)	Reserved for system use.

Column Name	Data Type	Description
UPDATES	CHAR(1)	Reserved for system use.
IBMREQD	CHAR(1)	Reserved for system use.
REMARKS	CHAR(78)	The table description.
DEFAULT	CHAR(1)	Reserved for system use.
KEYSEQ	INT	The column's numeric position in the primary key.
FOREIGNKEY	CHAR(1)	Reserved for system use.
FLDPROC	CHAR(1)	Reserved for system use.
LABEL	CHAR(30)	Reserved for system use.
ALIAS	CHAR(66)	Reserved for system use.
SEGNAME	CHAR(8)	Reserved for system use.
SEGTYPE	CHAR(4)	Reserved for system use.
SKEYS	INT	Reserved for system use.
PARENT	CHAR(8)	Reserved for system use.
INDEXED	CHAR(1)	Reserved for system use.
ACTUAL	CHAR(8)	Reserved for system use.
CODEPAGE	INT	Reserved for system use.
DBCSCODEPG	INT	Reserved for system use.
AVECOLLEN	INT	Reserved for system use.
TITLE	CHAR(78)	Alternate column title for reporting.
HELPMESSAGE	CHAR(78)	Help message for column.
USAGE	CHAR(8)	Usage format of column.
DIMEN_NAME	CHAR(66)	Dimension this column belongs to (OLAP data source).
DIMEN_ID	CHAR(4)	Reserved for system use.
DIMEN_LEVEL	CHAR(4)	Level in hierarchy of this column.

Column Name	Data Type	Description
RESTRICT	CHAR(2)	Shows type of description for this field.
FILENAME	CHAR(64)	File from which field originated.
REFERENCE	CHAR(66)	Extra description for dimension.
PROPERTY	CHAR(66)	Extra description for dimension.
XDEFAULT	CHAR(256)	Reserved for system use.
ACC_REL	INT	Type of accepts clause 0 EQ, 1 from, 2 TO, 3 FIND.
ACC_LENGTH	INT	True length of next column.
ACC_VALUE	CHAR(256)	Value in alpha format or FIND clause.

SYSKEYS (Full-Function and Hub Servers)

SYSKEYS contains one row for each key column in a data source identified to a server. A key column uniquely identifies a row in a data source.

Column Name	Data Type	Description
IXNAME	CHAR(18)	The first 18 characters of the column name.
IXCREATOR	CHAR(8)	The authorization ID of the owner of the synonym or data source.
TBNAME	CHAR(64)	The synonym or data source.
COLNAME	CHAR(66)	The full column name.
COLNO	INT	Reserved for system use.
COLSEQ	INT	Reserved for system use.
ORDERING	CHAR(1)	Specifies whether the key is sequenced in ascending or descending order: A=Ascending D=Descending
IBMREQD	CHAR(1)	Reserved for system use.
DBMS_CREATOR	CHAR(8)	The real database owner of the table.

SYSINDEX (Full-Function and Hub Servers)

SYSINDEX contains one row for each indexed column in a data source identified to a server.

Column Name	Data Type	Description
NAME	CHAR(66)	The name of the column.
CREATOR	CHAR(8)	The authorization ID of the owner of the synonym or data source that contains the column.
TBNAME	CHAR(64)	The synonym or data source that contains the column. **Note:** The default length for this field is 18, but can be changed to support up to 64 characters.
TBCREATOR	CHAR(8)	Same as CREATOR.
UNIQUERULE	CHAR(1)	Specifies valid entries: U=Unique entries only. D=Duplicate entries are allowed. P=Primary index.
COLCOUNT	INT	The number of fields in the index.
CLUSTERING	CHAR(1)	Reserved for system use.
CLUSTERED	CHAR(1)	Reserved for system use.
DBID	INT	Reserved for system use.
OBID	INT	Reserved for system use.
ISOBID	INT	Reserved for system use.
DBNAME	CHAR(8)	Reserved for system use.
INDEXSPACE	CHAR(8)	Reserved for system use.
FIRSTKEYCARD	INT	Reserved for system use.
FULLKEYCARD	INT	Reserved for system use.
NLEAF	INT	Reserved for system use.
NLEVELS	INT	Reserved for system use.
BPOOL	CHAR(8)	Reserved for system use.

Column Name	Data Type	Description
PGSIZE	INT	Reserved for system use.
ERASERULE	CHAR(1)	Reserved for system use.
DSETPASS	CHAR(8)	Reserved for system use.
CLOSERULE	CHAR(1)	Reserved for system use.
SPACE	INT	Reserved for system use.
IBMREQD	CHAR(1)	Reserved for system use.
CLUSTERRATIO	INT	Reserved for system use.
CREATEDBY	CHAR(8)	Reserved for system use.
COLNAMES	CHAR(256)	Reserved for system use.
IID	INT	Reserved for system use.
DBMS_CREATOR	CHAR(8)	The real database owner of the table.

SYSFILES (Full-Function and Hub Servers)

SYSFILES contains information on Master Files or stored procedures.

Column Name	Data Type	Description
FILENAME	CHAR(64)	Name of the file. For example, if you create a synonym for a data source using the Catalog Administrator, the name you give the synonym appears in this column.
LGNAME	CHAR(8)	Logical name of the location of FILENAME (for example, ddname).
PHNAME	CHAR(80)	Physical location of the file. This column contains the first 80 characters of the location; for locations that exceed 80 characters, additional columns are used. On MVS, this column is the name of the data set that contains the file.
PHNAME2	CHAR(80)	Continuation of the physical location of the file if required.

Column Name	Data Type	Description
PHNAME3	CHAR(80)	Continuation of the physical location of the file if required.
PHNAME4	CHAR(80)	Continuation of the physical location of the file if required.
PHNAME5	CHAR(80)	Continuation of the physical location of the file if required.
PHNAME6	CHAR(80)	Continuation of the physical location of the file if required.
PHNAME7	CHAR(32)	Continuation of the physical location of the file if required.
VERSION	INT	Version number of the file. Applies only to MVS.
MOD	INT	Modification number of the file. Applies only to MVS.
LINECNT	INT	Number of lines in the file.
DATE	CHAR(8)	Date of last change to the file, represented as DD/MM/YY.
TIME	CHAR(8)	Time of last change to the file, represented as HH.MM.SS.
USER ID	CHAR(100)	On MVS, the ID of the last user who changed the file.
SIZE	INT	On MVS, the number of lines in the file.

SYSRPC (Full-Function and Hub Servers)

SYSRPC contains one row for each cataloged stored procedure on a Hub Server, or one row for each local stored procedure on a Full-Function Server. It also describes execution parameters and answer set columns.

Column Name	Data Type	Description
RPC_NAME	CHAR(18)	Name of the procedure.
CREATOR	CHAR(8)	Procedure creator. The default is the current session ID.
LOCATION	CHAR(18)	Name of the server on which the procedure resides.
NUM_IPARMS	INT	Number of input parameters.
NUM_OPARMS	INT	Number of output parameters.
NUM_RESULTS	INT	Number of answer sets.
PID	INT	Reserved for system use.
RPC_DESC	CHAR(60)	Procedure description.
RPC_ALIAS	CHAR(18)	Name of the stored procedure on the subserver.
PARM_NAME	CHAR(18)	Parameter name.
TYPE	INT	Parameter type: 1=Input 3=Output
TYPE_NAME	CHAR(32)	Data type name.
DATA_TYPE	INT	Data type code.
LENGTH	INT	Length of column for data type CHAR.
PRECISION	INT	Length for numeric data type DECIMAL (packed or zoned).
NULLABLE	INT	Specifies whether the field contains a null value: 0=Not null 1=Null 2=Unknown

Column Name	Data Type	Description
SCALE	INT	Maximum number of digits to the right of the decimal.
RADIX	INT	Valid values are: 10=Digits for DECIMAL 2=Bits for FLOAT
PARM_ORDER	INT	Sequence number of the column in the answer set, or in the function call that executes the stored procedure.
PARM_DESC	CHAR(60)	Parameter description.

SYSCOLLN (Full-Function and Hub Servers)

SYSCOLLN contains one row for each collection of data sources.

Column Name	Date Type	Description
CNAME	CHAR(12)	Name of the collection.
EDASYNONYM	CHAR(8)	Alias for the collection (a shorter name for convenience).
EDAACCESS	CHAR(8)	Reserved for system use.
IDENTIFY	CHAR(1)	Reserved for system use.
REMARKS	CHAR(78)	Comments about the collection.

SYSCOLLT (Full-Function and Hub Servers)

SYSCOLLT contains one row for each data source in a collection. A data source can belong to more than one collection.

Column Name	Data Type	Description
CNAME	CHAR(12)	Name of the collection.
TBNAME	CHAR(64)	Name of the data source assigned to the collection.

CHAPTER 5

Resource Governor and Resource Analyzer Administration

Topics:

- Resource Governor Features
- Resource Governor/Resource Analyzer Operations
- Collecting Data With the Usage Monitor
- Setting Resource Limits (Thresholds)
- Building Rules
- Setting Governing On and Off
- Governing With the Usage Monitor
- Governing
- Administrative Databases
- Usage Monitoring Databases

Resource Governor and Resource Analyzer are optional components of the server environment. Resource Governor provides preemptive governing for requests issued to both relational and non-relational data sources. A data source can be either a physical relational table name or a Master File name. Resource Analyzer allows you to generate reports based on usage monitoring data.

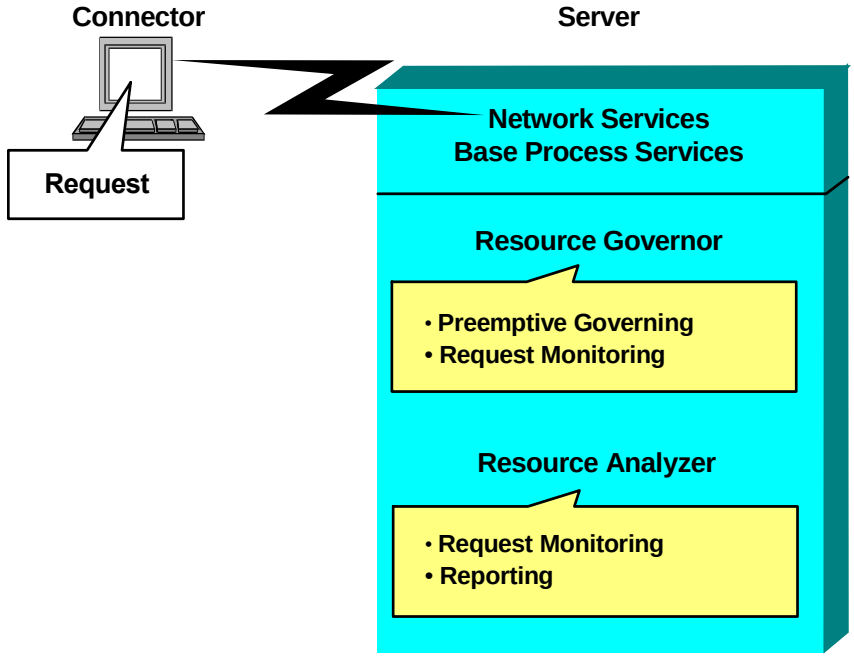
Resource Governor and Resource Analyzer use a Usage Monitor facility to gather statistics about how data is accessed and used. An administrator defines site-specific thresholds designating the amount of resources a request may use.

Based on this usage and threshold information, Resource Governor builds rules about how requests are to be governed against specific data sources. The governing facility uses the rules when inspecting each request, stopping any request that exceeds the predetermined resource thresholds.

The terms *usage monitoring* and *collecting* are used interchangeably.

Resource Governor Features

The following diagram illustrates Resource Governor/Resource Analyzer as it resides in the server environment.



To help you manage your site activity, Resource Governor controls data resources by placing resource limits on requests allowed to run. To establish this control, Resource Governor:

- Predicts resource usage, relative to the current threshold limits, before executing a request.
- Allows requests within acceptable usage limits to process.
- Prevents users from processing requests that exceed the specified limits.

Resource Governor/Resource Analyzer Operations

The following is a summary of the basic operations Resource Governor and Resource Analyzer undergo, and the steps you need to take to monitor server activity. A detailed explanation of each operation is provided later in this chapter.

Note: The first step, collecting usage monitoring data, and the last step, deleting data, are performed by both Resource Analyzer and Resource Governor. The other processes described in this chapter are for Resource Governor only. For more information on Resource Analyzer's administrative and reporting facilities, see the *Resource Analyzer Administrator's and User's Manual, Version 5*.

Operation	Description
Collect data about the system (Resource Governor and Resource Analyzer).	Turn usage monitoring on for your site. Resource Governor/ Resource Analyzer then records the requests and data about their resource usage. See *Collecting Data With the Usage Monitor* on page 5-7 for more information on usage monitoring.
Describe resource thresholds for various time periods (Resource Governor only).	Enter the relative resource thresholds to set limits on the amount of resources each request may use (amount of time and rows returned). Monitored resources include elapsed time and the number of result rows returned to the client. These thresholds help determine which requests can run during a specific time period while governing. See *Setting Resource Limits (Thresholds)* on page 5-12 for more information on thresholds.
Create rules (Resource Governor only).	Ask Resource Governor to induce and compile rules into a rules file. It then uses the rules in this rules file to evaluate each request and predict whether the request operates within the thresholds specified. See *Building Rules* on page 5-15 for more information on rules.
Turn Governing on (Resource Governor only).	Turn Governing on for each data source you want to monitor. See *Setting Governing On and Off* on page 5-48 for more information about turning governing on and off.

Operation	Description
Govern data source access (Resource Governor only).	Every time a request uses a monitored data source, Resource Governor predicts whether the request exceeds thresholds, based on the rules it has created. Only requests that are estimated to use less than or equal to the threshold amount of time and result rows can run. Requests that are estimated to exceed the thresholds are canceled. See *Governing* on page 5-49 for more information on governing.
Delete data (Resource Governor and Resource Analyzer).	Delete data from the Resource Governor/Resource Analyzer usage monitoring and administrative databases in order to reduce storage use. See *Maintaining the Usage Monitor Databases* on page 5-10 for more information on deleting data.

You can control these operations using the following remote procedures:

Procedure	Location of Script	Description
GKECOL	For MVS: EDARPC.DATA(GKECOLL) For UNIX and Windows NT/2000: EDACONF/BIN/GKECOLL.T3I	Sets usage monitoring on or off for a data source, including all sources.
GKEPARM	For MVS: EDARPC.DATA(GKEPARMS) For UNIX and Windows NT/2000: EDACONF/BIN/GKEPARMS.T3I	Inserts threshold values for a certain time of day and day of week.
GKERULE	For MVS: EDARPC.DATA(GKERULES) For UNIX and Windows NT/2000: EDACONF/BIN/GKERULES.T3I	Induces and compiles rules into an executable rules file (knowledge base) that can be loaded quickly into memory.
GKEGOV	For MVS: EDARPC.DATA(GKEGOVN) For UNIX and Windows NT/2000: EDACONF/BIN/GKEGOVN.T3I	Sets governing on or off for a data source.

Procedure	Location of Script	Description
GKEDEL	For MVS: EDARPC.DATA(GKEDELE) For UNIX and Windows NT/2000: EDACONF/BIN/GKEDELE.T3I	Enables you to maintain the Usage Monitor tables, by deleting Usage Monitor records no longer needed, to keep them representative of the site's current data access activity.

The following sections explain how to change control settings by specifying parameters in remote procedures.

MVS Only: These remote procedures are submitted through the *qualif*.EDACTL.DATA(GKEBAT) JCL, which is shown below:

```
//*
//* Supply a proper job card.
//*
//*********************************************************************
//**         Sample Resource Governor/Resource Analyzer Batch JOB    **
//**   Please copy this template before any modifications are done   **
//*********************************************************************
//*
//GKEBATCH   EXEC PGM=TSCOM3
//STEPLIB    DD  DISP=SHR,DSN=qualif.EDALIB.LOAD
//ERRORS     DD  DISP=SHR,DSN=qualif.EDAMSG.DATA
//EDASPROF   DD  DISP=SHR,DSN=qualif.EDAPROF.DATA(profile)
//*EDAPROF    DD  DISP=SHR,DSN=qualif.EDAPROF.DATA
//FOCECEC     DD  DISP=SHR,DSN=qualif.EDARPC.DATA
//ACCESS     DD  DISP=SHR,DSN=qualif.EDAAFD.DATA
//MASTER     DD  DISP=SHR,DSN=qualif.EDAMFD.DATA
//SMARTLIB   DD  DISP=SHR,DSN=qualif.SMARTLIB.DATA
//*
//*REMARK/DESCRIPTION: The following DD statement is used if      */
//*                  : EDA collection has been selected.         */
//*                  :                                           */
//*                  : Uncomment the DD statement and change the */
//*                  : DSN to point to the EDASU.DATA created for */
//*                  : the SU batch job(EDACTL.DATA(GKESU)).     */
//*
//*FOCSU01    DD  DISP=SHR,DSN=qualif.FOCSU.DATA
//*
```

```
//*****************************************************************
//**         TSCOM3 Script Output and Input Files                **
//*****************************************************************
//TRMOUT     DD  SYSOUT=*,DCB=(LRECL=133,BLKSIZE=133,RECFM=F)
//*****************************************************************/
//*REMARK/DESCRIPTION: The following DD statements will execute a */
//*                  : EDA script file located in EDACTL.DATA.    */
//*                  : The script will execute a procedure located*/
//*                  : in EDARPC.DATA. Any input parameter changes*/
//*                  : are placed in the script file pointed to   */
//*                  : by the TRMIN DD statement.                 */
//*                  :                                            */
//*                  : Uncomment the DD you wish to use. Only one */
//*                  : DD statement can be uncommented at a time. */
//*****************************************************************/
//*REMARK/DESCRIPTION: GKECOLL set collection ON or OFF for the
//*                  : specified object. This is also used to
//*                  : set GLOBAL collection ON or OFF.
//*
//*TRMIN      DD  DISP=SHR,DSN=qualif.EDACTL.DATA(GKECOLL)
//*
//*REMARK/DESCRIPTION: GKEPARMS set system parameters used to
//*                  : creating knowledge base rules
//*
//*TRMIN      DD  DISP=SHR,DSN=qualif.EDACTL.DATA(GKEPARMS)
//*
//*REMARK/DESCRIPTION: GKERULES creates rules using parameters
//*                  : set at installation or by the GKEPARMS
//*                  : procedure. The knowledge base name is also
//*                  : input to the GKEGOVN procedure.
//*
//*TRMIN      DD  DISP=SHR,DSN=qualif.EDACTL.DATA(GKERULES)
//*
//*REMARK/DESCRIPTION: GKEGOVN set governing ON or OFF for the
//*                  : specified object. This is also used to
//*                  : set the knowledge base name for the object.
//*
//*TRMIN      DD  DISP=SHR,DSN=qualif.EDACTL.DATA(GKEGOVN)
//*
//*REMARK/DESCRIPTION: GKEDELE is used to clean up system tables
//*                  : and/or collection data.
//*
//*TRMIN      DD  DISP=SHR,DSN=qualif.EDACTL.DATA(GKEDELE)
//
```

Uncomment the appropriate TRMIN DD statements associated with the task you want to invoke. For example, to set usage monitoring on or off for a specific data source, uncomment the following line:

```
//*TRMIN DD DISP=SHR, DNS=qualif.EDACTL.DATA(GKECOLL)
```

Update the script, GKECOLL, found in the *qualif*.EDACTL.DATA data set as described in *How to Set the Usage Monitor On and Off* on page 5-9.

Collecting Data With the Usage Monitor

The Usage Monitor facility accumulates information about requests and their actual resource usage. You can specify whether you want Resource Governor/Resource Analyzer to collect information about all requests in the system or only selected data sources. You start usage monitoring before you begin Resource Governor governing, in order to build up a representative sample of data on server activity. Resource Governor uses usage monitoring data to generate rules by which it later governs requests, deciding whether a request should be allowed to run. Resource Analyzer uses usage monitoring data to generate reports about site activity. Once you begin governing, you can choose to continue usage monitoring or turn it off.

The remote procedure GKECOL contains the parameters that enable you to control usage monitoring. See *How to Set the Usage Monitor On and Off* on page 5-9 for more information on GKECOL. For more details about usage monitoring, see the *Resource Governor User's Manual,* or the *Resource Analyzer Administrator's and User's Manual*.

Note: Information can be collected for all SQL statements or TABLE/TABLEF, MATCH FILE, GRAPH, and MAINTAIN requests. Governing does not occur for any SQL statements that use INSERT, UPDATE, DELETE, or DROP.

The Usage Monitor

The Usage Monitor acquires information about the system's pattern of resource usage by logging statistics about requests issued at the site. This information includes:

- SQL request or TABLE request
- MATCH FILE request
- MAINTAIN request
- User ID
- Request time and date
- Data source queried
- Elapsed and CPU time in seconds
- Row count

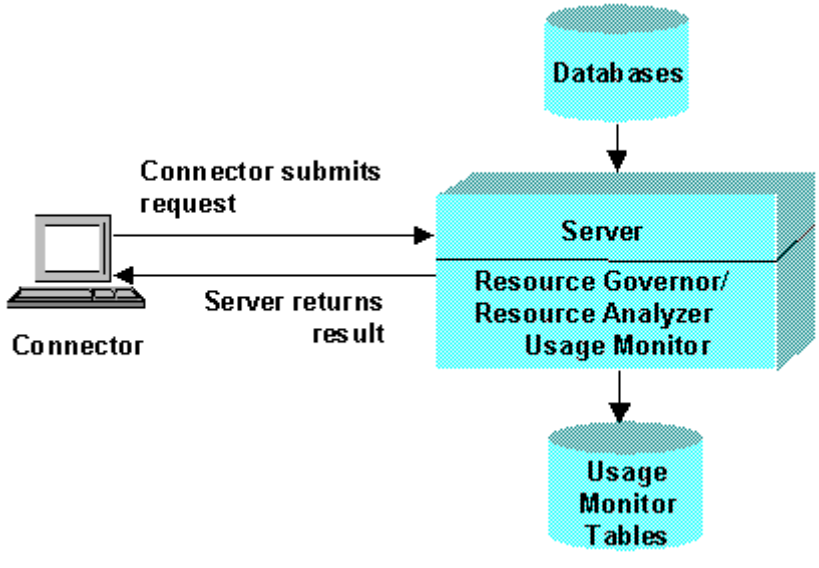
This information is stored in the Usage Monitor databases (see *Usage Monitoring Databases* on page 5-61 for a detailed description). Once a representative sample of data has been collected, it is later used to generate rules that are used to determine whether a request may continue (see *Building Rules* on page 5-15 for more information).

You can turn sample usage monitoring on or off for any or all queried data sources at any time. For example, to exclude certain data sources from Resource Governor/Resource Analyzer's control, you can set global usage monitoring on and turn it off for an individual data source. See *How to Set the Usage Monitor On and Off* on page 5-9 for more information on starting and stopping usage monitoring.

Once you turn on Resource Governor's governing facility, you can choose to continue usage monitoring for future reference (if you later want to generate new rules), or you can turn usage monitoring off. It is possible, however, that while Resource Governor is governing, usage monitoring can occur even if you have turned the usage monitoring setting off for a particular data source used in a request. This can happen if:

- The Governor could not estimate correctly. When this occurs, a complete set of usage monitor data is recorded, including resource usage. Thus, the next time a rule file is created, the rules can take this kind of scenario into account.
- The request is canceled. In this case, only the SQL statement or TABLE request and governing statistics are saved, since there is no actual resource usage information to record.

The Usage Monitor process is illustrated below:



Procedure How to Set the Usage Monitor On and Off

To begin usage monitoring of data, perform one of the following:

- Turn on global Usage Monitor for all data objects accessed under Resource Governor/ Resource Analyzer's control.
- Turn on Usage Monitor for specific data sources.

The remote procedure GKECOL creates the Usage Monitor/Governor file, GKTABLE. GKTABLE is a flat table that records which data sources are being monitored and, in the case of Resource Governor, governed. GKTABLE is loaded by Resource Governor/Resource Analyzer at request time. This file contains the information needed to properly inspect any requests that pass through the server. It must be rebuilt every time usage monitoring information is updated.

Parameters in GKECOL enable you to turn on Usage Monitor globally or for specific data sources.

Syntax How to Turn On Usage Monitor Globally or for Specific Data Sources

```
EX GKECOL TBNAME='tbname', SOURCE='source', COLLECT={'ON'|'OFF'},
```

where:

tbname

Designates the data source name as used by queries created by connectors. For example, to collect information with the Usage Monitor about queries using a DB2 table and Passthru, specify the DB2 creator and table name, such as CREATOR.TABLE.

A three-part name can be used if the target RDBMS allows three qualifiers, such as LOCATION.CREATOR.TABLE. If the data source is accessed through the catalog, list the Master File name that is recorded in the catalog, such as EMPLOYEE.

To set global collection on, specify @GLOBAL in place of the data source name.

source

Specifies the actual source of the table access. If TBNAME is a catalog name, the source is the server.

If TBNAME is a physical table or database referenced through Passthru to a relational data service, the source is that service's engine name.

Note: SOURCE is ignored if TBNAME='@GLOBAL'.

COLLECT

Starts or stops usage monitoring for the specified data sources or turns it off. Specify ON to turn the Usage Monitor on, or OFF to turn it off.

Notes:

- You can reproduce the GKECOL procedure any number of times in order to run it for different data sources (TBNAMEs).
- If most data sources are to be monitored, you can selectively disable usage monitoring for individual databases by setting the Usage Monitor off and setting global collection on. The following example shows how to use GKECOL to accomplish this.

```
EX GKECOL TBNAME='@GLOBAL',              SOURCE='EDA', COLLECT='ON'
EX GKECOL TBNAME='EXCEPTION1',           SOURCE='EDA', COLLECT='OFF'
EX GKECOL TBNAME='CREATOR2.EXCEPTION2', SOURCE='DB2', COLLECT='OFF'
```

  As this example also illustrates, you can reproduce the lines with EX GKECOL using different parameter values as many times as desired.
- The GKECOL procedure also updates the SMCONTROL database. See *Administrative Databases* on page 5-52 for a description of the SMCONTROL database.

Maintaining the Usage Monitor Databases

Passing queries through the Usage Monitor facility generates a representative sample of resources used for the data sources targeted for governing. You need to maintain the Usage Monitor databases to keep them representative of the site's current reporting activity. Resource Governor/Resource Analyzer provides the remote procedure GKEDEL for this purpose. The COLLECT option deletes from the Resource Governor/Resource Analyzer databases any records that match the object name (TBNAME) and date range passed to it. The SYSTEM option deletes records that match TBNAME in the system databases.

Syntax How to Maintain the Usage Monitor Databases With GKEDEL

```
EX GKEDEL TBNAME='tbname', SOURCE='source',
          TYPE='type',
          EDATE='yyyymmdd', LDATE='yyyymmdd',
```

where:

tbname

Designates the data source name as used by queries created by connectors. For example, to collect information with the Usage Monitor facility about queries using a DB2 table and Passthru, specify the DB2 creator and table name, such as CREATOR.TABLE.

A three-part name can be used if the target RDBMS allows three qualifiers, such as LOCATION.CREATOR.TABLE. If the data source is accessed through the catalog, list the Master File name that is recorded in the catalog, such as EMPLOYEE.

source

Specifies the actual source of the table access. If TBNAME is a catalog name, the source is the server.

If TBNAME is a physical table or database referenced through Passthru to a relational data service, the source is the engine name of the service.

type

Indicates if either administrative or Usage Monitor data is to be deleted or if both are to be deleted. Possible values are:

indicates data from the Usage Monitor databases will be deleted.

SYSTEM indicates that administrative or system database data pertaining to the data source name listed in TBNAME will be deleted.

BOTH indicates that both types of data will be deleted.

EDATE

Indicates the earliest date of Usage Monitor data to delete. This value must be in the format

yyyymmdd

where:

yyyy

Is the four-digit year.

mm

Is the two-digit month.

dd

Is the two-digit day of the month.

LDATE

Designates the latest date of Usage Monitor data to delete. It must be in the same format as EDATE.

Note:

- The EDATE and LDATE parameters apply to COLLECT or BOTH; they *do not* apply to SYSTEM.
- You can reproduce the GKEDEL procedure any number of times in order to run it for different data sources (TBNAMEs).

Resource Analyzer Administration and Reporting

While usage monitoring is performed by both Resource Governor and Resource Analyzer, the remaining administrative tasks (setting thresholds, creating rules, governing) described in this chapter apply only to Resource Governor.

Setting Resource Limits (Thresholds)

Before Resource Governor can create the rules that establish limits on the amount of resources each request can use, you first need to provide information about permissible resource thresholds at your site. Resource Governor then uses these thresholds along with the already collected usage monitoring data to build a rule file of rules about how requests are to be governed against specific data sources. Based on these rules, the Resource Governor governing facility then decides if a request can execute or not. For more information, see *Building Rules* on page 5-15.

You use the remote procedure GKEPARM to assign a maximum threshold for two parameters used for generating rules:

- The number of result rows.
- The time used to return the result rows (the number assigned to time represents elapsed time or CPU seconds).

You can set separate thresholds according to time of day by specifying separate starting and ending times and using the name values.

GKEPARM enables you to store a variety of criteria in the SMPRMTRS table.

Syntax How to Set Resource Limits

```
EX GKEPARM NAME='shiftname', STRTTIME='hhmm', ENDTIME='hhmm'
           STRTDATE='mmdd', ENDDATE='mmdd'
           CPUTIME=3600, ELAPTIME=1400, ROWS=10, USED='1',
           MONDAY='1', TUESDAY='1', WEDNESDAY='1', THURSDAY='1',
           FRIDAY='1', SATURDAY='0', SUNDAY='0', TYPE={'S'|'D'}
```

where:

shiftname

Designates a character string identifier that indicates when certain threshold values are in effect for a given time period. This value may be any string consisting of eight characters or less. The same name may be used for multiple shifts provided that the dates and times are different.

An acceptable value is DEFAULT. The threshold values for the parameters TIME and ROWS are used as a default for this entry if no other entries exist for the current date or time.

`STRTTIME`

Designates the start time for the thresholds to take effect. This must be in the format

hhmm

where:

hh

Is the two-digit hour.

mm

Is the two-digit minute.

`ENDTIME`

Designates the end time for the thresholds to be in effect. Use the same time formats as shown for STRTTIME above.

`CPUTIME`

Specifies the time threshold based on CPU seconds. CPU seconds is the measurement that the server CPU is consuming to satisfy the request.

`ELAPTIME`

Specifies the time threshold based on elapsed time. Elapsed time is the physical clock time in seconds that the request used from start to finish.

`ROWS`

Indicates the result row threshold. This threshold is the maximum number of result rows or records that can be returned to the client. For example, if the administrator does not want result sets with more than 10,000 rows returned to any client, the threshold should be set to 10,000.

`USED`

Indicates whether shift is currently active. 1= active shift; 2= inactive shift.

`MONDAY`

Indicates a daily type of shift if active. 1= active shift; 0= inactive shift.

`TUESDAY`

Indicates a daily type of shift if active. 1= active shift; 0= inactive shift.

`WEDNESDAY`

Indicates a daily type of shift if active. 1= active shift; 0= inactive shift.

`THURSDAY`

Indicates a daily type of shift if active. 1= active shift; 0= inactive shift.

`FRIDAY`

Indicates a daily type of shift if active. 1= active shift; 0= inactive shift.

SATURDAY

Indicates a daily type of shift if active. 1= active shift; 0= inactive shift.

SUNDAY

Indicates a daily type of shift if active. 1= active shift; 0= inactive shift.

TYPE='S'

Indicates a shift using dates and times.

TYPE='*D*'

Indicates a shift using daily indicators and time ranges. (Dates are not used).

STRTDATE

Designates the start date for the thresholds to take effect. This value must be in the format

mmdd

where:

mm

Is the two-digit month.

dd

Is the two-digit day.

ENDDATE

Designates the end date for the thresholds to be in effect. Use the same date formats as shown for STRTDATE above.

Note:

- You can reproduce the GKEPARM stored procedure any number of times within this procedure, allowing you to save multiple shifts and thresholds in Resource Governor's SMPRMTRS database. The parameter values saved in the SMPRMTRS database are then used along with usage monitoring data to create rules for governing.
- During installation, SMPRMTRS is initialized with a set of default thresholds, as shown below.

```
VALUES ('DEFAULT', '0000', '2359', '0101', '1231', 3600, 1440, 100000,
10000, '1', '0', '0', '0', '0', '0', '0', 'S');
```

This set of thresholds, called DEFAULT, allows for 3,600 seconds or one hour of time, or 100,000 result rows returned to the client by governed requests. See *Administrative Databases* on page 5-52 for more information.

Building Rules

Before Resource Governor can begin governing queries, you must first generate the rules that Resource Governor uses to determine if a request exceeds the designated permissible thresholds. Resource Governor uses two kinds of rules:

- **Automatically generated rules (data rules).** Resource Governor creates these rules based on previously collected usage monitoring data and the thresholds specified in GKEPARM. These Resource Governor-created rules estimate a query's CPU time and number of returned rows.
- **Custom rules.** You create these rules to cover more specific circumstances and criteria than those covered by Resource Governor-created rules.

When a request is issued, the Governor looks at the request and uses the specified rules (data, custom, or both) to determine whether the request should run.

The remote procedure GKERULE compiles these rules into a rule file (or knowledge base), pulling in either or both the Resource Governor-generated rules and the custom rules created in GKECR.

Syntax How to Compile Rules Into a Rule File

```
EX GKERULE TBNAME='tbname', SOURCE='source', KBNAME='kbname', CUSTOM='cusrules',
KNBTYPE='knbtype', EDATE='yyyymmdd', LDATE='yyyymmdd', TIME='time',
```

where:

tbname

Designates the data source name as used by queries created by connectors. For example, to collect information with the Usage Monitor about queries using a DB2 table and Passthru, specify the DB2 creator and table name, such as CREATOR.TABLE.

A three-part name can be used if the target RDBMS allows three qualifiers, such as LOCATION.CREATOR.TABLE. If the data source is accessed through the catalog, list the one identifier that is recorded in the catalog, such as EMPLOYEE.

source

Specifies the actual source of the table access. If TBNAME is a catalog name, the source is the server.

If TBNAME is a physical table or database referenced through Passthru to a relational data service, the source is the engine name of the service.

kbname

Designates the name of the rule file, created by this procedure that is used by any requests made to the specified data source. This rule file name does not take affect until you run the GKEGOV procedure to indicate which rule file to use for governing.

cusrules

'GKECR' or the name of the custom rule produced using the Custom Rule Wizard.

knbtype

Indicates the types of rules you want to create. Possible values are:

```
B=Both; C=Custom only; D=Data only.
```

EDATE

Indicates the earliest date the Usage Monitor data was recorded. This date is used for selecting records from the Usage Monitor tables. The date must be in the format

yyyymmdd

where:

yyyy

Is the four-digit year.

mm

Is the two-digit month.

dd

Is the two-digit day.

LDATE

Indicates the latest date the Usage Monitor data was recorded. Use the same date formats as shown for EDATE= above.

time

Designates whether elapsed or CPU time should be used as governing criteria when comparing to the time threshold in a rule file. Both are stored in seconds.

Note:

- You can reproduce the GKERULE procedure any number of times in order to run it for different data sources (TBNAMEs).
- A rule file name does not get used until its name is entered in and governing turned on with the GKEGOV procedure. See *Setting Governing On and Off* on page 5-48 for information about using GKEGOV.
- The GKERULE procedure updates the SMKBASE and SMBRL system databases. See *Administrative Databases* on page 5-52 for descriptions of the information stored in these databases.

Keeping Rules Current

There are a number of reasons for which the rules currently in use may no longer be appropriate for one or more of the governed data sources:

- Significant permanent changes were made in the characteristics of the data source or in the types of requests used. (For example, a large number of records were added or deleted.)
- The site's Usage Monitor information is not representative of current requests. Even if the data source has not changed, you can discover that some types of recurring requests were not recorded or used in the date parameter range.
- The Resource Governor administrator made changes in thresholds that are desired in the existing governing setup. Thus, GKERULE no longer reflects what is in GKEPARM.
- Seasonal differences are present in the Usage Monitor data.

  To handle seasonal activity, generate a separate set of rules (by reproducing the GKERULE procedure) for a season using the date parameters. When a season changes, create new rules with the appropriate date ranges of Usage Monitor data with GKERULE. As long as you use different names when creating rules, multiple rule files can be stored. You can change the active name for each data source under Resource Governor's control through GKEGOV. See *Setting Governing On and Off* on page 5-48.

Custom Rules and Messages

The rules generated by Resource Governor are based on usage monitoring data and the kinds of requests that have been monitored before governing is turned on. You may, however, know of certain conditions contained in requests to the server that should never be allowed to run. If these conditions never occur during the representative sampling of usage monitoring data, however, they will not become part of Resource Governor's rules. Likewise, you may know of certain conditions under which all queries should be allowed to run, such as a user ID with greater privileges. You can create custom rules to check for these kinds of conditions and generate cancellation messages, or allow beyond-threshold queries to run.

Customization allows Resource Governor to govern immediately on conditions that you know are unacceptable or that you know should never be canceled. It also lets you designate more specific parameters beyond simply the time and row number thresholds specified in GKEPARM. For example, you can have Resource Governor govern based on the particular data source being queried, the user ID submitting the request, or even a particular field within the request. Custom rules allow for flexibility upon configuration of Resource Governor or changing Information System conditions at your site. Create these custom rules using the Resource Governor Custom Rule Builder using Business Rule Language (BRL). See *Using Business Rule Language (BRL)* on page 5-20 for more details on BRL.

Creating Custom Rules With GKECR (No Graphical User Interface)

The Resource Governor Custom Rule Builder is a Graphical User Interface that allows the administrator, using a wizard and a software development kit, to create custom rules. If you choose not to use this interface, you may edit the file called GKECR to produce custom rules. GKECR can be found in the following locations:

Platform	File Location
MVS	*qualif*.EDACTL.DATA data set
UNIX	$EDAHOME/etc directory
VM	Production disk
Windows NT	%EDAHOME% \etc directory

You may want to customize rules to provide certain governing exemptions under unique situations. GKECR provides several templates of custom rules that you can edit to customize for your particular needs.

To create your own custom rules, edit the rules found in GKECR using BRL syntax as described in *Using Business Rule Language (BRL)* on page 5-20. When you customize the rules in GKECR, make sure to remove the comment characters. The GKERULE procedure always incorporates the rules in GKECR, so that every time you rebuild the rule files with GKERULE, the custom rules in GKECR are executed. However, since GKECR contains comment characters when delivered, GKECR rules will not be executed until these characters are removed.

Rebuilding Rule Files After Custom Rule Changes

For query governing, you can use only Resource Governor's automatically generated rules, a combination of these with your own custom rules, or your custom rules alone. The remote procedure GKEKNB lets you control this selection. GKEKNB can be found in the following locations:

Platform	File Location
MVS	*qualif*.EDARPC.DATA data set
UNIX and WINDOWS NT	$EDAHOME/catalog directory
VM	Production disk

You can use GKEKNB in two ways:

- If you have never run GKERULE, and therefore do not yet have any rule files, you can run GKEKNB to create a rule file of only your custom rules from GKECR.
- If you have already run GKERULE and have at least one rule file in place, you can run GKEKNB to add new custom rules without regenerating the Resource Governor-created rules. Use the same name as a pre-existing rule file so you can update it with custom rules without incurring the expense of regenerating all the Resource Governor rules with GKERULE.

The Resource Governor-created rules are stored in the Resource Governor database, SMPRL, and are retrieved when you update an existing rule file by using its name when running GKEKNB.

Syntax How to Update an Existing Rule File

```
EX GKEKNB TBNAME='tbname',
KBNAME='knbname',
CUSTOM='custfile'
```

where:

tbname

Is the data source name used by the queries created by connectors.

knbname

Is the name of the rule file that is used by any requests made to the specific data source referenced in TBNAME.

custfile

Is the name of the custom rule file. This is the value for SMKNBNAME in the table SMPRL. Use GKECR when not generating custom rules using the Custom Rule Builder.

Using Business Rule Language (BRL)

Resource Governor lets you customize rules and messages for a particular user or situation using a language called Business Rule Language (BRL). BRL allows IF/THEN testing on certain information available when the rules execute. Your own cancel message can be included within the rules you create by using special message variables.

BRL is a non-procedural, high-level application development language that allows you to develop sophisticated programs with less effort than conventional programming languages. BRL rules consist of three parts: the rule name, which serves as a comment or description, and is not syntactically necessary to the rule; a supporting condition (antecedent) or procedure statement; and a conclusion.

The procedure and conclusion are expressed through IF-THEN statements. When a rule is executed, Resource Governor determines whether these IF-THEN statements lead to a specified goal. Every BRL rule file, or knowledge base, must have at least one goal statement describing a conclusion that can be reached by your rules and that affects governing decisions, deciding whether the query is to be canceled or run. The final goal statement pre-established for all Resource Governor rules is *DBA Rules Concluded*, but you may also specify additional sub-goals within your customized rule file.

Resource Governor uses a process called *backward chaining*, which involves starting with the final goal and working *backward* through the subgoals expressed within the rules to arrive at the final goal. Since BRL is non-procedural, the order of rules is not important—Resource Governor will execute all the rules as many times as necessary until the conditions of the query have been determined—and rules can be added at any time, anywhere in the rule file.

The following section describes the kinds of information that can be contained within BRL IF-THEN statements.

BRL Factual Information

BRL is capable of representing several different types of information within a single rule file or rule:

- Simple factual statements.
- Numeric data.
- String-type data.
- Attribute-value (A-V) associations.

These facts are described in the following sections. These sections give examples based on the following rule:

```
RULE        Storm Conditions
IF          There are high winds present
AND         Barometric pressure < 28.8
AND         The kind of clouds present are cumulus
AND         month = "March"
OR          month = "April"
THEN        There is a storm present
```

Reference Simple Facts

A simple fact is a straightforward expression that requires only a true or false answer. An example of a simple fact is “There are high winds present.”

Reference Numeric Facts

Numeric data lets you compare values, create computations, and assign numeric data to variables. BRL recognizes that you are specifying a numeric data type by the presence of one of the reserved relational, numerical, or assignment operators within a supporting condition (antecedent, or IF statement).

Relational Operators	<	Less than
	>	Greater than
	<=	Less than or equal to
	>=	Greater than or equal to
	< >	Not equal
	=	Equal
Assignment Operator	:=	

When you are using numeric facts, all variables must be declared as a NUMERIC type.

An example of a numeric fact is, “Barometric pressure < 28.8.”

Reference String Facts

With string facts, you can pass string information to external programs, construct customized messages, write customized external files, and so on. The target variable must be pre-declared as a STRING type. An example of a string fact is, month = "March".

Reference Attribute-Value Facts

An attribute-value association is a statement in which an attribute is described by a particular value. In this way, it resembles a simple fact, except that a simple fact has only one value associated with the attribute, while the attribute-value association uses variables to allow for a variety of values. An example of an attribute-value fact is, "The kind of clouds present ARE cumulus," since there are several different values that could be substituted for "cumulus" after the "ARE."

Reference Backward Chaining

The sample BRL in this section is a purely hypothetical, non-real world example that provides a simple illustration of how the backward chaining process works. For this example, there are four possible final goals, each representing an action you can take to get to the theater:

- Walk to the theater.
- Take a coat and walk to the theater.
- Take a cab to the theater.
- Drive your car to the theater.

The sample BRL includes a number of rules that determine which final goal will be chosen, based on the kind of input given.

```
RULE   Determining to Drive (alternative #1)              !  Rule 1
IF     Distance > 5
THEN   Means := "Drive"

RULE   Determining to Drive (alternative #2)              !  Rule 2
IF     Distance > 1
AND    Time til curtain < 15
THEN   Means := "Drive"
```

```
RULE   Determining whether to walk                          !  Rule 3
IF     Distance > 1
AND    Distance < 15
AND    Time til curtain > 15
AND    Nice out
OR     Nasty out
THEN   Means := "Walk"

RULE   Determining travel action (alternative #1)           !  Rule 4
IF     Means = "Drive"
AND    Location = "Downtown"
THEN   Take a cab to theater

RULE   Determining trace action (alternative #2)            !  Rule 5
IF     Means = "Drive"
AND NOT Location = "Downtown"
THEN   Drive you car to theater

RULE   Determining if Nasty out                             !  Rule 6
IF     Raining out
OR     Snowing out
OR     Hailing out
THEN   Nasty out

RULE   To see if it is raining out now                      !  Rule 7
IF     Relative Humidity > 80
AND    Barometric pressure IS falling
THEN   Raining out

RULE   That it is snowing out                               !  Rule 8
IF     Relative Humidity > 70
AND    Temperature < 34
THEN   Snowing out

RULE   To determine if it is Hailing out                    !  Rule 9
IF     Relative Humidity > 70
AND    Temperature <= 32
THEN   Hailing out

RULE   for Nice out                                         !  Rule 10
IF     Relative Humidity <= 50
AND    Relative Humidity >= 20
AND    Temperature >= 65
AND    Temperature <= 86
AND    Sunny
THEN   Nice out

RULE   Determining travel action (alternative #3)           !  Rule 11
IF     Means = "Walk"
AND    Nasty out
THEN   Take coat and walk to theater
```

```
RULE   Sunny now                                              !  Rule 12
IF     cloud count <= 3
THEN   Sunny

RULE   Determining travel action (last alternative)           !  Rule 13
IF     Means = "Walk"
AND    Nice out
THEN   Walk to theater
END
```

The following steps outline the backward chaining process for this BRL:

1. The action of how to get to the theater is established as four separate goals. (These goals are established before you begin constructing the BRL syntax.)
2. Rules 4, 5, 11, and 13 conclude about this action. Again, note that because of BRL's non-procedural design, these do not need to be in any particular order—the concluding rules could just as easily be the first four or the last four in the syntax.
3. Rule 4 concludes that the travel action is "Take a cab to the theater." This is the first rule that will be attempted to be fired, since it is the first one in the syntax whose conclusion is one of the final goals. Its premise is true only if Means is "Drive" and location is "Downtown"; thus obtaining the value for Means is the first subgoal to pursue.
4. Rules 1, 2, and 3 conclude about Means. Rule 1 is tested first. If it fails, Rule 2 is tested. If, for example, the distance is 2 miles and the show starts in 10 minutes, then Rule 2's premise succeeds. Rule 2 concludes that the Means is "Drive." Now Resource Governor has a choice to make. It could take this single value for Means and go back to Rule 4, or it could continue searching for other rules that conclude about Means. If the rule is an attribute-value type fact, meaning that it could take on more than one value, then the search for additional rules should continue. In this example, the search continues. Rule 3 is tested, but it fails because of "Time til curtain." The supporting condition, "Nice out," is not pursued, but it would have been if the "Time til curtain" numeric fact supporting condition was >= 15 minutes.
5. No other rules conclude about Means. The subgoal is complete and Resource Governor returns to focus on Rule 4.
6. The second supporting condition of Rule 4, Location, comes under inspection. If it is true, that is, if Location is "Downtown," then Rule 4 succeeds (fires), and the goal "Take a cab to theater" is concluded. Since more that one goal exists in this BRL, Rules 5, 11, and 13 are pursued. Rule 5 fails for Location; Rules 11 and 13 fail for Means. The BRL chaining ends.

Try running this BRL for other Distance, Location, and Time til curtain inputs to see how the rules are pursued and fired based on your various test inputs.

Example Using BRL

This example of BRL is more complex than the previous one, showing actual syntax for a server for MVS. The BRL shown here allows the DBA to issue a message to end users when specific databases are inaccessible because of maintenance or system unavailability. It also illustrates how custom rules work together with Resource Governor's automatically generated rules.

For this example, the DBA would have to type the list of one or more unavailable databases into a sequential data set (in this case, SYS1.UNAVAIL.DBASES.LIST). In addition, the DBA would enter a custom message into the data set SYS1.MESSGE.*databasename*. This BRL contains examples of how to detect full-table scans and select *s, two operations that may be expensive to run. It also illustrates how to establish user overrides and Cartesian product join detection, as well as a number of other capabilities.

```
RULE Conclude DBA Rules Main                                    !Rule1
IF Database Unavailable
OR Nocancel
OR Omnipotent User Override
OR Cartesian Product Join
OR Full Table Scan
OR Selected All Columns
OR Too Many Joins in affect
OR Valid Cancel
THEN DBA Rules Concluded
!

RULE Check If Database Unavailable                              !Rule 2
IF  Initialize and Allocate
AND Unavailable
AND Open custom message file
AND Read and Write Messages
AND Deallocate
THEN Database Unavailable
AND  Run := "F"
AND Reason := "DB  Down"
!

RULE Initialize and Allocate                                    !Rule 3
THEN Initialize and Allocate
AND ioresult := 0
AND DDN 1 := "UNAVAIL"
AND Dynam Data := "ALLOC FILE UNAVAIL"
AND Dynam Data 1 := " SHR REUSE DS PGMEIB.UNAVAIL.DBASES.LIST"
AND ACTIVATE %DYNAM
DATA Dynam Data
DATA Dynam Data 1
!
```

```
RULE Unavailable                                                !Rule 4
IF Get Table Names
AND Read List
AND ioresult = 0
THEN Unavailable
!
RULE Find Unavailable Database                                  !Rule 5
IF Tablename 1 = Record
OR Tablename 2 = Record
OR Tablename 3 = Record
OR Tablename 4 = Record
OR Tablename 5 = Record
THEN Database Found
!
RULE Open custom message file                                   !Rule 6
THEN Open custom message file
AND DDN 2 := Record
AND Dynam Data := "ALLOC FILE "
AND ACTIVATE %CONCAT
DATA Dynam Data
DATA DDN 2
AND Dynam Data 1 := " SHR REUSE DS PGMEIB.MESSAGE."
AND ACTIVATE %CONCAT
DATA Dynam Data 1
DATA DDN 2
AND ACTIVATE %DYNAM
DATA Dynam Data
DATA Dynam Data 1
!
```

```
RULE Read and Write Messages                          !Rule 7
ACTIVATE %GET
DATA DDN 2
DATA Record
DATA ioresult
IF  message line 1
AND message line 2
AND message line 3
AND message line 4
AND message line 5
AND ioresult <> 0
THEN Read and Write Messages
ELSE FORGET message line 1
AND  FORGET message line 2
AND  FORGET message line 3
AND  FORGET message line 4
AND  FORGET message line 5
AND Lineptr := Lineptr + 1
AND LOOP
!

RULE to build message line 1                          !Rule 8
IF Lineptr = 1
THEN message line 1
AND Message1 := Record
ELSE message line 1
!

RULE to build message line 2                          !Rule 9
IF Lineptr = 2
THEN message line 2
AND Message2 := Record
ELSE message line 2
!

RULE to build message line 3                          !Rule 10
IF Lineptr = 3
THEN message line 3
AND Message3 := Record
ELSE message line 3
!

RULE to build message line 4                          !Rule 11
IF Lineptr = 4
THEN message line 4
AND Message4 := Record
ELSE message line 4
!
```

```
RULE to build message line 5                                    !Rule 12
IF Lineptr = 5
THEN message line 5
AND Message5 := Record
ELSE message line 5
!
RULE Read List                                                  !Rule 13
ACTIVATE %GET
DATA DDN 1
DATA Record
DATA ioresult
ACTIVATE %TRIM
DATA Record
DATA right
IF Database Found
OR ioresult <> 0
THEN Read List
ELSE FORGET Database Found
AND LOOP
!
RULE Get Table Names                                            !Rule 14
ACTIVATE %TABLES
READ
DATA Table IS WHAT
IF  Make string 1
OR  Make string 2
OR  Make string 3
THEN Get Table Names
!
RULE Make string 1                                              !Rule 15
IF Table IS CAR
THEN Make string 1
AND  Tablename 1 := "CAR"
!
RULE Make string 2                                              !Rule 16
IF Table IS PGMEIB.COMPANY
THEN Make string 2
AND  Tablename 2 := "COMPANY"
!
RULE Make string 3                                              !Rule 17
IF Table IS PGMEIB.EMPLOYEES
THEN Make string 3
AND  Tablename 3 := "EMPLOYEES"
!
```

```
RULE Valid Cancel                                          !Rule 18
IF Run <> "T"
THEN Valid Cancel
AND Message1 := "Execution cannot be allowed as query"
AND Message2 := "will potentially consume excessive resources."
!

!  rule for Disable governing after 6 PM and before 8 AM

RULE Nocancel                                              !Rule 19
IF Run <> "T
AND  Hours and minutes >= "1800"
AND  Hours and minutes <= "0800"
THEN Nocancel
AND  Run := "T"
!

RULE To allow Omnipotent users                             !Rule 20
IF Userid = "CFO     "
OR Userid = "PRESIDNT"
OR Userid = "VICEPRES"
THEN Omnipotent User Override
AND Run := "T"
!

!  No WHERE conditions so a FULL table scan is imminent

RULE No Where Conditions Found                             !Rule 21
IF Number of relations = 0
THEN Full Table Scan
AND Run := "F"
AND Reason := "FullScan"
AND Message1 := "Please add a WHERE condition to your request"
AND Message2 := "to avoid a full table scan and excessive"
AND Message3 := "resource consumption."
!

RULE Selected All Columns                                  !Rule 22
IF Select asterisk
THEN Selected All Columns
AND Run := "F"
AND Reason := "SELECT*"
AND Message1 := "Please do not choose EVERY columns in this"
AND Message2 := "particularly wide database.  Instead"
AND Message3 := "choose specific column names in your query."
!
```

```
!  Check to see if the user is issuing a Cartesian Product with
!  the join of two databases
RULE Cartesian Product Join                                    !Rule 23
IF Number of tables = 2
AND Number of relations = 0
THEN  Cartesian Product Join
AND Run := "F"
AND Reason := "CartProd"
AND Message1 := "Apply a WHERE clause to constrain this"
AND Message2 := "join. Your request is a Cartesian Product"
AND Message3 := "and will return n X m rows. "
!
!
!  The use of many joins can be a resource hog
!
RULE Too Many Joins in affect                                  !Rule 24
IF Number of tables >= 4
THEN  Too Many Joins in affect
AND Run := "F"
AND Reason := ">=4joins"
AND Message1 := "You are using too many joined data objects."
AND Message2 := "For efficiency, extract data to temp files"
AND Message3 := "and join the temp files to persistent databases."
!
! Deallocate the DDNAME for unavail list and the message file
RULE Deallocate                                                !Rule 25
THEN Deallocate
AND Dynam Data := "FREE FILE"
AND Dynam Data 1 := DDN 1
AND ACTIVATE %DYNAM
DATA Dynam Data
DATA Dynam Data 1
AND Dynam Data 2 := DDN 2
AND ACTIVATE %DYNAM
DATA Dynam Data
DATA Dynam Data 2
!
RULE Conclude DBA Rules Fallthrough Rule
THEN DBA Rules Concluded
```

Rule 1 is the main rule and is inspected first. If any one of the OR conditions *fires* (that is, is true), then the rule is completed and no further OR conditions of Rule 1 are inspected. The next rule, Rule 2, must fire four additional rules in order to confirm that a database is unavailable, as well as to send a message to the end user. You can follow the rules to their conclusion by reading Rules 3 through 7. Rule 7 reads a flat file and sets up messaging output (Rules 8 through 12). Rule 13 reads the list of currently unavailable databases from a flat file. Rule 18 messages the user when the Resource Governor has determined using Resource Governor's automatically generated rules that the query exceeds the user input threshold. Rules 19 through 24 address specific query characteristics. For example, Rule 23 checks to see if the request is joining two tables without an IF/WHERE condition (a Cartesian product join), which is a potentially costly and inefficient action. Rule 25 performs clean-up operations.

Reference

BRL Keywords

You can add customized rules as required by your site. All BRL keywords must be written in uppercase.

Variable Declaration:

Keyword	Description
`ATTRIBUTE`	Declares the type of a variable to be an attribute-value pair. Each pair is treated as a simple Boolean. There are two types of attributes: single-valued (only one attribute-value pair can be true at any moment) and multiple-valued (one, some, or all the attribute-value pairs can be true simultaneously). The default is single-valued. (See MULTI in the *Control Element Selectors* table.)
`NUMERIC`	Declares the type of a variable to be numeric.
`SIMPLEFACT`	Declares the type of a variable to be logical, that is, true or false.
`STRING`	Declares the type of a variable to be a character string.

Control Element Selectors:

Keyword	Description
EXHAUSTIVE	Normally, once a SIMPLEFACT is determined to be true or false, no other attempts to determine its value will be performed and no other rules that conclude it will be pursued. If EXHAUSTIVE is specified for the SIMPLEFACT, all rules that conclude it will be pursued even if its value has already been determined.
MULTI	Defines an ATTRIBUTE as possibly having multiple values.

Rule Declaration:

Keyword	Description
ACTIVATE	Allows execution of external routines or pre-defined internal functions. See *Internal Functions* on page 5-39 for a list of executable ACTIVATE routines.
AND	Serves as a logical operator for joining variables or a sequencing operator when it connects a series of actions or statements. In premises containing both AND and OR, statements are evaluated from the top to the bottom of the rule, in order of appearance.
ARE	The keywords IS, ARE, and the symbol \ can be used interchangeably. (See the keyword IS.)
DATA	When used with READ, causes a variable to be assigned a value that is read from memory in the server. In conjunction with WRITE, DATA causes the value of a fact to be written to memory in the server. The purpose of DATA is to transfer the value of a variable between a rule file and the server or functions invoked through the ACTIVATE keyword. Each DATA keyword must be on its own line in the rule file. The value of a variable read in by DATA goes into the session context and is useable from then on.
END	Marks the end of a rule file and signals the termination of the text of BRL. Omitting END will result in a compiler error.
IF	Introduces the premise of a rule. There can only be one IF per rule. Multiple premises can be tested and joined by AND or OR.

Keyword	Description
INIT	Enters values into the session context when a rule file is first executed. The purpose of INIT is to assign values needed when processing begins. Variables initialized with INIT are available before any rules execute.
IS	Binds an ATTRIBUTE variable to a value. In the premise of a rule, the IS value is interpreted as a test to determine whether the attribute-value pair is a true statement. In a conclusion, the IS value assigns a value of true to the statement. The keywords IS, ARE, and the symbol \ can be used interchangeably.
NOT	Negates the true value of a variable. In the premise of a rule, a statement beginning with NOT is true if the variable following NOT is false and vice versa. In the conclusion of a rule, NOT enters the following variable into the session with the value false if the rule executes.
OR	The logical operator for connecting variables in the premise of a rule. OR makes it possible to combine rules having the same conclusion into a single rule. In premises containing both AND and OR, statements are evaluated from the top to the bottom of the rule, in order of appearance.
READ	Causes values from the following DATA keywords to be assigned to the variables listed with DATA.
RULE	Specifies the name of a BRL rule. Every rule must begin with the reserved word RULE followed by the name of the rule. Rule names do not have to be distinct. All rules must contain the keyword THEN followed by a variable that is a SIMPLEFACT or ATTRIBUTE.
THEN	Introduces the primary conclusion of a rule. Every rule must have the keyword THEN. All the BRL statements in the primary conclusion are executed when the premise of the rule is true. The first statement following THEN must be a SIMPLEFACT or ATTRIBUTE type of variable. Numeric, string, and procedural statements are not permitted immediately following THEN although they are allowed elsewhere in the conclusion.
TITLE	Marks the beginning of a BRL rule file. The first line of the rule file is the TITLE line. The length of the title cannot exceed 60 characters.
WHAT	Used with IS, ARE, or the symbol \ and the keyword DATA to indicate that multiple values are to be returned or sent for an ATTRIBUTE (as in DATA attribute IS WHAT).

Keyword	Description
`WRITE`	Causes values from the following DATA keywords to be passed back to the server or ACTIVATE function. A WRITE keyword must be followed by one or more DATA keywords.
`$ [file]`	The include operator, $, specifies a file to be included in a rule file. The name following the $ symbol will be merged into the rule file at compile time. Files included with the $ operator cannot contain other include files. The $ operator must be located in column one.
`!`	The comment operator marks the beginning of a comment in a BRL source file.
`"  "`	Quotation marks designate a string literal. Literals must be enclosed in double quotation marks to permit the use of reserved BRL symbols and words and leading and trailing blanks when assigning a value to a variable, and to differentiate a literal value for a variable from a variable name.
`:`	The continuation operator indicates that a line in a BRL source file is continued on the next line.
`:=`	The assignment operator assigns the value on the right side of the := symbol to the variable on the left side (this operator is used with NUMERIC and STRING variables).
`\`	The keywords IS, ARE, and the symbol \ can be used interchangeably. (See the keyword IS.)
`Arithmetic Operators`	Used in performing calculations involving numeric expressions. `( )` Parenthesis. `*` Multiplication. `/` Division. `+` Addition. `-` Subtraction.

Keyword	Description
Relational Operators	Relational operators are not allowed in conclusions of rules. Strings or numerics can be compared. = Equal to. < > Not equal to. < Less than. > Greater than. <= Less than or equal to. >= Greater than or equal to.

Reference **Pre-defined Variables**

Variables are case and white-space sensitive. For example, TIME is a different variable than time. The following sections list all the variables provided in a rule file. A MULTI can have from 1 to 256 values.

These rules apply to all of the variables listed.

- NUMERIC variables are standard double-precision floating-point data types.
- SIMPLEFACT variables are always true or false.
- STRING variables can contain up to 80 characters.

In an assignment, a premise, or a conclusion in a rule, variables are always referenced by themselves or with the NOT operator. For example, IF *simple1* or IF NOT *simple1*.

NUMERIC:

Variable	Description
Current Time Threshold	The threshold in effect for elapsed or CPU time in seconds for the current date and time.
Current Rows Threshold	The threshold in effect for the maximum size of the requested result set.
Time Threshold	The threshold estimated by the rule file for elapsed or CPU seconds.
Rows Threshold	The threshold estimated by the rule file for maximum size of result set.

Variable	Description
Rule Number	An arbitrary number assigned to a rule to enable the reference of what rule did or did not cancel a request.
Day of Week	The day of the week indicated as 0 - 6, where 0 is Sunday and 6 is Saturday.
Extra Number	Defined for use by custom rules.
Number of unions	The number of UNION statements contained in the request.
Number of union alls	The number of UNION ALL statements contained in the request.
Number of tables	The number of tables or data sources contained in the request.
Number of relations	The number of relational clauses contained in the request.
Number of group bys	The number of GROUP BY columns contained in the request.
Number of order bys	The number of ORDER BY columns contained in the request.
Number of functions	The number of functions contained in the request.
Number of columns	The number of columns contained in the request or records inserted into the table SMCOLUMNS. It is primarily the number of columns selected except when an asterisk is used. In that case, the number of columns is one.

SIMPLEFACT:

Variable	Description
Correlated query	Used by the automated rules created with GKERULE, and can be used in custom rules as well.
Select asterisk	Used by the automated rules created with GKERULE, and can be used in custom rules as well.
Bad or not condition	Used with the %ORNOTERR function. Identifies whether an invalid logical combination of not/or conditions are specified in the current SQL WHERE clause.
Distinct columns	Used with the %DISTINCT function. Specifies whether the SQL DISTINCT parameter was used in the current query.

STRING:

Variable	Description
`User ID`	The ID of the client connected to the server.
`Reason`	An eight-character string that contains a reason why the Governor decided what it did. This value is stored in the Usage Monitor table, SMGOVERN.
`Current Date`	The current date in the form YYYYMMDD.
`Hours and minutes`	The current time in the form HHMM.
`Message1`	The first string defined that can contain a custom message. It is always returned to the server. It is used if there is a value assigned to it and overrides the standard cancellation message.
`Message2`	The second string defined that can contain a custom message. It is always returned to the server. It is used if there is a value assigned to it and Message1.
`Message3`	The third string defined that can contain a custom message. It is always returned to the server. It is used if there is a value assigned to it and the preceding message strings.
`Message4`	The fourth string defined that can contain a custom message. It is always returned to the server. It is used if there is a value assigned to it and the preceding message strings.
`Message5`	The fifth string defined that can contain a custom message. It is always returned to the server. It is used if there is a value assigned to it and the preceding message strings.
`Format`	A string value unused at this time. Available for any use in custom rules.
`Extra String`	A string value unused at this time. Available for any use in custom rules.
`Run`	The indicator for whether the decision is to run or cancel the request. It is initialized to `T`. Custom rules can set it to `F` if desired, which causes the request to cancel.

ATTRIBUTE:

Variable	Description
`Table Name`	Used by the automated rules created with GKERULE.
`Relation`	Used by the automated rules created with GKERULE.
`Table`	Used with the %TABLES function. Its values are the table names specified in the SQL FROM clause of the current query.
`Column Name`	Used with the %COLUMNS function. Its values are the column names used in the SQL SELECT clause of the query.

`Relational`	Used with the %RELOPCOL function. Its values are the relational statements from the SQL WHERE clause with the right-hand side specified as LITERAL or FIELD. For example,
	`WHERE (SMQUERY.SMKEY = SMFROMS.SMKEY) AND (SMQUERY.SMDATE > '19960101')`
	would become the values:
	`SMQUERY.SMKEY.EQ.FIELD`
	and
	`SMQUERY.SMDATE.GT.LITERAL`

`Left relational column`	Used with the %LEFTREL function. Its values are the left-hand side of the relational statements specified in the SQL WHERE clause. Using the previous example, the values would be:
	`SMQUERY.SMKEY`
	and
	`SMQUERY.SMDATE`

`Right relational column`	Used with the %RIGHTREL function. Its values are the right-hand side of the relational statements specified in the SQL WHERE clause when a field is specified. Using the previous example, the only value used would be: `SMQUERY.SMKEY`
`Group by column`	Used with the %GROUPBY function. Its values are the column names specified in the SQL GROUP BY clause of the current query.

Variable	Description
`Order by column`	Used with the %ORDERBY function. Its values are the column names specified in the SQL ORDER BY clause of the current query.
`Function`	Used with the %FUNCTION function. Its values are the function names specified in the SQL WHERE clause of the current query.
`Function Column`	Used with the %FUNCOLS function. Its values are the function and column names specified in the SQL WHERE clause of the current query.
`Relational Operator`	Used with the %RELOPS function. Its values are the relational operators specified in the SQL WHERE clause. Using the previous example, the values EQ.FIELD and GT.LITERAL would be used.

`Relational Statement`	Used with the %RELATION function. Its values are the entire relational statements from the SQL WHERE clause. Using the previous example, the values used would be
	`SMQUERY.SMKEY.EQ.creator.SMFROMS.SMKEY`
	and
	`SMQUERY.SMDATA.GT.19960101`

`Table Relations`	Used with the %RELTABS function. Its values are the table names relation from the SQL WHERE clause. Using the previous example, the value used would be: `SMQUERY.EQ.creator.SMFROMS`

Reference **Internal Functions**

The following are internal functions executable as ACTIVATE routines in the rules.

Function	Description
`%COLUMNS`	Returns column names.
`%DISTINCT`	Determines if DISTINCT columns were requested.
`%FUNCOLS`	Returns functions with column names.
`%FUNCTION`	Returns function names.

Function	Description
%GROUPBY	Returns group names.
%LEFTREL	Returns relational columns (left side).
%ORDERBY	Returns ordered columns.
%ORNOTERR	Invalid combination of OR and NOT logic.
%RELATION	Returns the relational statements.
%RELOPCOL	Returns the relational column/operator combinations and determines whether they compare against a literal or a field.
%RELOPS	Returns relational operator AND/OR with LITERAL or FIELD.
%RELTABS	Returns relational conditions between tables.
%RIGHTREL	Returns relational columns (right side).
%TABLES	Returns table names.

Examples in the following function descriptions are based on this sample SQL request:

```
SELECT T1.COLUMN1,T2.COLUMN2,AVG(T2.COLUMN4),MAX(T2.COLUMN5)
FROM CREATOR.TABLE1 T1,CREATOR.TABLE2 T2
WHERE (T1.INDEX1 = T2.INDEX2) AND (T1.COLUMN3 > '19951231')
GROUP BY T1.COLUMN1,T2.COLUMN2
ORDER BY T1.COLUMN1,T2.COLUMN2;
```

Reference **%COLUMNS**

The %COLUMNS function returns all the column names specified in the SELECT clause.

Use:

```
RULE Get column names
ACTIVATE %COLUMNS
DATA Column name IS WHAT
THEN Column names retrieved
```

Based on the sample SQL, the following attribute-value pairs are true:

```
Column name IS CREATOR.TABLE1.COLUMN1
Column name IS CREATOR.TABLE2.COLUMN2
```

Reference %DISTINCT

The %DISTINCT function returns true if DISTINCT is specified in the SELECT clause; otherwise, it returns false.

Use:

```
RULE Get distinct
ACTIVATE %DISTINCT
DATA Distinct columns
THEN Distinct determined
```

Based on the sample SQL, the SIMPLEFACT Distinct columns are false. Distinct columns would be true if the sample had been:

```
SELECT DISTINCT T1.COLUMN1,...
```

Reference %FUNCOLS

The %FUNCOLS function returns all the functions with column names specified in the SELECT clause.

Use:

```
RULE Get functions with columns
ACTIVATE %FUNCOLS
DATA Function column IS WHAT
THEN Functions with columns retrieved
```

Based on the sample SQL, the following attribute-value pairs are true:

```
Function column IS AVG.CREATOR.TABLE2.COLUMN4
Function column IS MAX.CREATOR.TABLE2.COLUMN5
```

Reference %FUNCTION

The %FUNCTION function returns all the function names specified in the SELECT clause.

Use:

```
RULE Get functions
ACTIVATE %FUNCTION
DATA Function name IS WHAT
THEN Functions retrieved
```

Based on the sample SQL, the following attribute-value pairs are true:

```
Function name IS AVG
Function name IS MAX
```

Reference %GROUPBY

The %GROUPBY function returns all the column names specified in the GROUP BY clause.

Use:

```
RULE Get group bys
ACTIVATE %GROUPBY
DATA Group by column IS WHAT
THEN Group bys retrieved
```

Based on the sample SQL, the following attribute-value pairs are true:

```
Group by column IS CREATOR.TABLE1.COLUMN1
Group by column IS CREATOR.TABLE2.COLUMN2
```

Reference %LEFTREL

The %LEFTREL function returns the left-hand side of the relational condition as specified in the WHERE clause.

Use:

```
RULE Get left relational column
ACTIVATE %LEFTREL
DATA Left relational column IS WHAT
THEN Left relational column retrieved
```

Based on the sample SQL, the following attribute-value pairs are true:

```
Left relational column IS CREATOR.TABLE1.INDEX1
Left relational column IS CREATOR.TABLE1.COLUMN3
```

Reference %ORDERBY

The %ORDERBY function returns all the column names specified in the ORDER BY clause.

Use:

```
RULE Get group bys
ACTIVATE %ORDERBY
DATA Order by column IS WHAT
THEN Order bys retrieved
```

Based on the sample SQL, the following attribute-value pairs are true:

```
Order by column IS CREATOR.TABLE1.COLUMN1
Order by column IS CREATOR.TABLE2.COLUMN2
```

Reference %ORNOTERR

The %ORNOTERR function returns true if a faulty OR/NOT condition is specified in a WHERE clause.

Use:

```
RULE Determine faulty relational condition
ACTIVATE %ORNOTERR
DATA Bad or not condition
THEN Faulty relational determined
```

Based on the sample SQL, the SIMPLEFACT Bad or not condition are false. If the sample had been

```
SELECT T1.COLUMN1
FROM CREATOR.TABLE1 T1
WHERE (NOT T1.COLUMN9 = 'RED' OR NOT T1.COLUMN9 = 'BLUE');
```

Then, when COLUMN9 is RED, it is not BLUE, so the OR makes the condition true; and when COLUMN9 is BLUE, it is not RED, so the OR still makes the condition true; and when COLUMN9 is neither RED nor BLUE, the condition is still true. The effect is the same as coding SELECT T1.COLUMN1 without a WHERE clause.

In this case, the Bad or not condition is true.

Reference %RELATION

The %RELATION function returns all the relational statements specified in the WHERE clause.

Use:

```
RULE Get relational statements
ACTIVATE %RELATION
DATA Relational statement IS WHAT
THEN Relational statements retrieved
```

Based on the sample SQL, the following attribute-value pairs are true:

```
Relational statement IS CREATOR.TABLE1.INDEX1.EQ.CREATOR.TABLE2.INDEX2
Relational statement IS CREATOR.TABLE1.COLUMN3.GT.19951231
```

Reference %RELOPCOL

The %RELOPCOL function returns all the relational column/operator combinations and determines whether they compare against a literal or a field as specified in the WHERE clause.

Use:

```
RULE Get relational column and operator
ACTIVATE %RELOPCOL
DATA Relational IS WHAT
THEN Relational column and operator retrieved
```

Based on the sample SQL, the following attribute-value pairs are true:

```
Relational IS CREATOR.TABLE1.INDEX1.EQ.FIELD
Relational IS CREATOR.TABLE1.COLUMN3.GT.LITERAL
```

Reference %RELOPS

The %RELOPS function returns all the relational operators and determines whether they compare against a literal or a field as specified in the WHERE clause.

Use:

```
RULE Get relational operator
ACTIVATE %RELOPS
DATA Relational operator IS WHAT
THEN Relational operator retrieved
```

Based on the sample SQL, the following attribute-value pairs are true:

```
Relational operator IS EQ.FIELD
Relational operator IS GT.LITERAL
```

Reference %RELTABS

The %RELTABS function returns all the relational conditions between tables as specified in the WHERE clause.

Use:

```
RULE Get relational table conditions
ACTIVATE %RELTABS
DATA Table relations ARE WHAT
THEN Relational table conditions retrieved
```

Based on the sample SQL, the following attribute-value pair are true:

```
Relational statement IS CREATOR.TABLE1.EQ.CREATOR.TABLE2
```

Reference %RIGHTREL

The %RIGHTREL function returns the right-hand side of the relational condition when it references a field as specified in the WHERE clause.

Use:

```
RULE Get right relational column
ACTIVATE %RIGHTREL
DATA Right relational column IS WHAT
THEN Right relational column retrieved
```

Based on the sample SQL, the following attribute-value pairs are true:

```
Right relational column IS CREATOR.TABLE2.INDEX2
```

Reference %TABLES

The %TABLES function returns all the table names specified in the FROM clause.

Use:

```
RULE Get table names
ACTIVATE %TABLES
DATA Table IS WHAT
THEN table names retrieved
```

Based on the sample SQL, the following attribute-value pairs are true:

```
Table IS CREATOR.TABLE1
Table IS CREATOR.TABLE2
```

Example Customized Rules

The following examples suggest more uses of custom rules. The goal of every rule is DBA Rules Concluded. The THEN statement should not change to ensure that the Governor finishes its conclusion properly.

Example 1:

This example illustrates the custom rule that would make additional information available to the rule file. All of the internal functions called with the ACTIVATE keyword initializes the ATTRIBUTE variables listed with the DATA statements. This allows you to write additional custom rules.

```
! Process internal functions
RULE Get information about the query
ACTIVATE %TABLES
DATA Table IS WHAT
ACTIVATE %COLUMNS
DATA Column name IS WHAT
ACTIVATE %DISTINCT
DATA Distinct columns
ACTIVATE %LEFTREL
DATA Left relational column IS WHAT
ACTIVATE %RIGHTREL
DATA Right relational column IS WHAT
ACTIVATE %RELOPCOL
DATA Relational IS WHAT
ACTIVATE %GROUPBY
DATA Group by column IS WHAT
ACTIVATE %ORDERBY
DATA Order by column IS WHAT
ACTIVATE %FUNCTION
DATA Function name IS WHAT
ACTIVATE %FUNCOLS
DATA Function column IS WHAT
ACTIVATE %RELOPS
DATA Relational operator IS WHAT
ACTIVATE %RELATION
DATA Relational statement IS WHAT
ACTIVATE %RELTABS
DATA Table relations ARE WHAT
ACTIVATE %ORNOTERR
DATA Bad or not condition
THEN All functions processed
```

Example 2:

This example illustrates how to check for one or more table names used in the original SQL request. For example, this rule file might have been built for TABLE2, and you know you never want TABLE1 used with TABLE2.

```
! Assuming this rule file is for TABLE2, do not allow joins to TABLE1
RULE Check table name
IF All functions processed
AND Table IS TABLE1
THEN DBA Rules Concluded
AND Run := "F"
AND Reason := "TABLE"
AND Rule Number := 9999
```

Example 3:

This rule example assures that if an equality condition exists on an indexed column in a WHERE clause, the request runs.

```
RULE Check Relation
IF All functions processed
AND Relational IS TABLE1.COLUMN1.EQ.LITERAL
THEN DBA Rules Concluded
AND Run := "T"
AND Reason := "INDEX"
```

Example 4:

This rule example cancels all requests using a SELECT * and no WHERE clauses.

```
RULE Check for Asterisk
IF Select asterisk
AND Number of Relations = 0
THEN DBA Rules Concluded
AND Run := "F"
AND Reason := "SELECT*"
AND Rule Number := 9998
AND Message1 := "Please add a WHERE clause to your SELECT statement."
AND Message2 := "You selected every row in the table."
AND Message3 :="5 Message variables are available."
```

Example 5:

The following rule example ensures no requests are canceled.

```
RULE Do not cancel any queries
THEN DBA Rules Concluded
AND Run := "T"
```

Setting Governing On and Off

After generating estimation rules for a data source, you can ask Resource Governor to begin governing it. The remote procedure GKEGOV enables you to set governing on for a data source or data sources. See *Building Rules* on page 5-15 for more information on compiling a rule file of rules.

Syntax How to Set Governing On for a Data Source

```
EX GKEGOV TBNAME='tbname', SOURCE='source', KBNAME='knbname', govern,
```

where:

tbname

Designates the data source name as used by queries created by connectors. For example, to collect information with the Usage Monitor facility about queries using a DB2 table and Passthru, specify the DB2 creator and table name, such as CREATOR.TABLE.

A three-part name can be used if the target RDBMS allows three qualifiers, such as LOCATION.CREATOR.TABLE. If the data source is accessed through the catalog, list the Master File name that is recorded in the catalog, such as EMPLOYEE.

source

Specifies the actual source of the table access. If TBNAME is a catalog name, the source is the server.

If TBNAME is a physical table or database referenced through Passthru to a relational data service, the source is that service's engine name.

knbname

Designates the name of the rule file that contains the rules used for governing requests made against the specified data source. See *Building Rules* on page 5-15 for more information on compiling a rule file of rules. If you are turning governing off, you can leave this name blank.

govern

Turns governing on and off. Specify ON to turn governing on, OFF to turn it off, or ADVISE to run the Governor in ADVISE mode. See *Using ADVISE* on page 5-50 for more information on this feature.

Note:

- You can reproduce the GKEGOV procedure any number of times in order to run it for different data sources (TBNAMEs) and parameters.
- If separate rule files are desired for seasonal use, you can change the rule file name by putting a different rule file name in the parameter list. If the rule file you specify in the parameter list does not exist, you must use GKERULE to create it.
- There is no facility for global governing. Each data source must be entered individually into Resource Governor using GKEGOV.

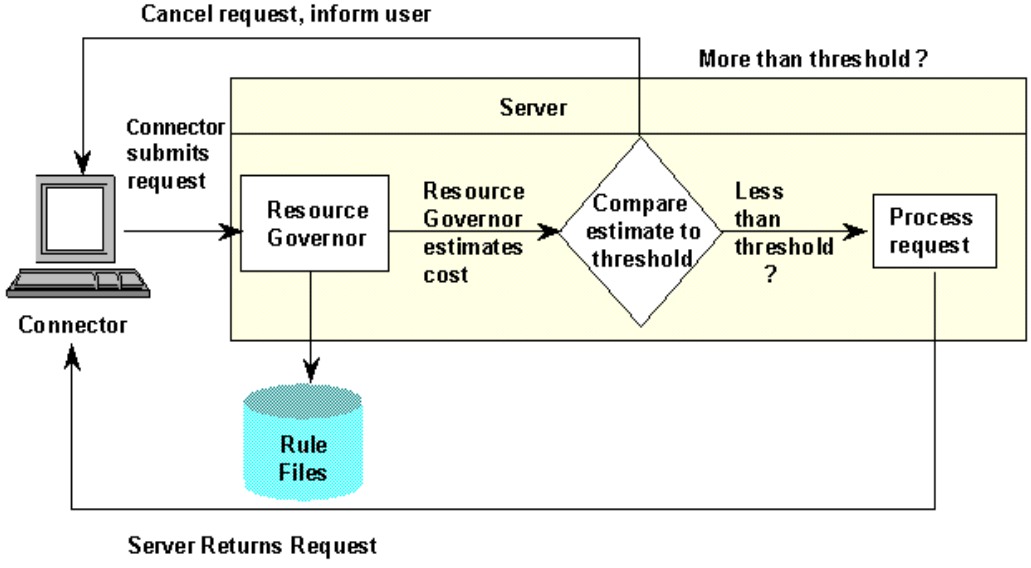
Governing

When a user submits a request against any data source monitored by Resource Governor, the governing facility performs high-speed resource usage estimates at run time, using rules it previously generated, which are stored in rule files. The Governor decides whether to run or cancel each request and, depending on the decision, one of the following actions occurs:

- If the decision is to Run, the request is allowed to execute.
- If the decision is to Cancel, the request is terminated, and a cancellation message is returned to the client.

In addition to the amount of resources each request potentially uses, the Governor considers the time of day, the date, and detailed characteristics of the request. The Governor can be turned on or off for any data source at any time using the remote procedure GKEGOV. See *Setting Governing On and Off* on page 5-48 for more information on GKEGOV.

The governing process is illustrated below:



Using ADVISE

ADVISE is a feature of Resource Governor that enables the System Administrator to ease Resource Governor into the production environment. When the governing mode is ADVISE, it indicates in the Usage Monitor data which request would have been canceled had governing been set on. The SMCOLLECT column of the SMQUERY database contains a value of 3, indicating that Usage Monitoring occurred while ADVISE was on. ADVISE gives you the ability to collect data with the Usage Monitor about a request that would normally have been canceled by the Governor. With this information, you can then decide if the threshold limits in effect are correct, or if governing should be started. Also, a message is sent back to the client indicating that the request would have been canceled.

Governing With the Usage Monitor

The following chart indicates what data is recorded by Resource Governor under what conditions. Some databases are populated depending on the request characteristics.

	Configurations				
Collection Databases	**Usage Monitoring only**	**Global collection**	**Govern and could not estimate**	**Govern with cancel**	**Govern with ADVISE**
SMQUERY	X	X	X	X	X
SMREQUESTS	X	X	X	X	X
SMFROMS	X	X	X		
SMCOLUMNS	X	X	X		
SMRELATIONS	X	X	X		
SMBYS	X	X	X		
SMFUNCTIONS	X	X	X		
SMGOVERN			X	X	X
SMRPCS	X	X	X	X	X

Administrative Databases

This section lists the data definitions that make up the Resource Analyzer administrative databases and provides an explanation of the column values.

SMCONTROL Database (SMCNTRL.MAS)

Keeps track of which data sources are monitored, governed, and have rules applied. This is the main monitoring and governing control database. It is updated every time the administrator either turns on or off collection for either Global Monitoring or individual object monitoring or governing.

Column	Value	Description
SMNAME	Alphanumeric, length=66	The SMNAME can be used for relational or non-relational data types. If it contains a relational database name, it must be fully qualified. For example, there are three identifiers contained in a complete name: 'LOCATION.CREATOR.TABLE'
SMSOURCE	Alphanumeric, length=8	The data type can be a DBMS accessed via SQL passthru, such as DB2; or the data type can be EDA, indicating that a server has access to the data type listed in its catalog. See your Server manual for a list of valid engine names.
SMKNBNAME	Alphanumeric, length=8	The rule file name used for governing.
SMCOLLECT	Alphanumeric, length=1	Indicates whether Resource Analyzer should record Usage Monitor data about any request using SMTBNAME. 0=Off 1=On
SMGOVERN	Alphanumeric, length=1	Indicates whether Resource Analyzer should govern any request using SMTBNAME. 0=Off 1=On
SMADVISE	Alphanumeric, length=1	Indicates whether Resource Analyzer should advise on canceling any request using SMTBNAME. 0=Off 1=On

SMRPCS Database (SMRPCS.MAS)

Contains information on any stored procedure or focexec execution if RPC monitoring is enabled.

Column	Value	Description
SMRPCKEY	Alphanumeric, length=40	Key for a single remote procedure.
SMRPCNUM	Integer	Remote procedure sequence in an execution chain.
SMRPCNAME	Alphanumeric, length=67	Long name of an executed remote procedure.
WFRPCNAME	Alphanumeric, length=66	Long name of an executed WebFOCUS remote procedure.
SMDATE	Alphanumeric, length=8	The date of the RPC run in format '*yyyymmdd*'.
SMTIME	Alphanumeric, length=6	The start time of the RPC run.
SMUSERID	Alphanumeric, length=30	The user identifier that made the connection to the Reporting Server.
SMCPUTIME	Integer	The total CPU time for the RPC in milliseconds.
SMELAPTIME	Integer	The total wall clock time in seconds.
SMIOS	Integer	The total IO for this RPC.
SMROWS	Integer	The total rows returned to the client.
SMROWWIDTH	Integer	The byte count of the widest row returned.
SMCONNADDR	Alphanumeric, length=32	The connection address.
SMCONNTYPE	Integer	The connection type. 1=TCP 2=SNA 3=IPX

Column	Value	Description
SMRPCNUM	Integer	The parent SMRPCNUM. 0=Root RPC
SMJOINFLD	Alphanumeric, length=1	Join field, the element needed to join this database to the RCAPRMS database.
SABANDWIDTH	Integer	The computed value of SMROWS * SMROWWIDTH.
SAMO	Alphanumeric, length=2	The numeric value of the month the request was executed.
SAYEAR	Alphanumeric, length=4	The numeric value of the year the request was executed.
SAMONTH	Alphanumeric, length=9	The character name of the month the request was executed.
SACONNTYPE	Alphanumeric, length=7	Indicates connection method used when the request was executed: 1=TCP 2=SNA 3=IPX 0=Unknown
SACPUTIME	Integer	The total CPU time for the RPC (in seconds).

SMKBASE Database (SMKBASE.MAS)

This is the database that is updated during the Build Rules process. It contains information on all Knowledge Base files created for the data objects.

Note: This database is created during installation and configuration. It is only used with Resource Governor.

SMPRL Database (SMPRL.MAS)

Column	Value	Description
SMNAME	Alphanumeric, length=66	The SMNAME can be used for relational or non-relational data types. If it contains a relational table name, it must be qualified. For example, there are three identifiers contained in a complete name: 'LOCATION.CREATOR.TABLE'
SMSOURCE	Alphanumeric, length=8	The data type can be a DBMS accessed using SQL Passthru, such as DB2; or the data type can be the server, indicating that a server has access to the data type listed in its catalog.
SMKNBNAME	Alphanumeric, length=8	The rule file name used for governing.
SMSHIFTNAME	Alphanumeric, length=8	The name for a time shift. Each rule file occurrence has at least the DEFAULT shift associated with.
SMSTARTDATE	Alphanumeric, length=4	The start date for a specific shift in the format: '*MMDD*' For example, if the shift is a holiday, such as Christmas, then the SMSTARTDATE= '1225'.
SMENDDATE	Alphanumeric, length=4	The end date for a specific shift in the format: '*MMDD*' For example, if the shift is a holiday, such as Christmas, then the SMENDDATE= '1225'.
SMSTARTTIME	Alphanumeric, length=4	The start time for the thresholds to take effect. The format is '*hhmm*,' which designates the hours and minutes.
SMENDTIME	Alphanumeric, length=4	The end time for the thresholds to take effect. The format is '*hhmm*,' which designates the hours and minutes.
SMCUSTOM	Alphanumeric, length=8	The name of the custom rule file.

Column	Value	Description
SMTIMETYPE	Alphanumeric, length=8	The time parameter passed to GKERULE or GKEKNB that indicates either CPU or ELAPSED time.
SMCREATED	Alphanumeric, length=8	The date/time that the rule file was created, in the format '*YYYYMMDD*'.
SMFIRSTDATE	Alphanumeric, length=8	The effective start date of the Usage Monitor data used, in the format '*YYYYMMDD*'.
SMLASTDATE	Alphanumeric, length=8	The effective end date of the Usage Monitor data used, in the format '*YYYYMMDD*'.
SMMONDAY	Alphanumeric, length=1	A flag that indicates whether this day of the week is used for this shift. 0=Do not use this day of the week for the shift. 1=Use this day of the week for the shift. **Note:** Used only when SMSHIFTYPE=D.
SMTUESDAY	Alphanumeric, length=1	A flag that indicates whether this day of the week is used for this shift. 0=Do not use this day of the week for the shift. 1=Use this day of the week for the shift. **Note:** Used only when SMSHIFTYPE=D.
SMWEDNESDAY	Alphanumeric, length=1	A flag that indicates whether this day of the week is used for this shift. 0=Do not use this day of the week for the shift. 1=Use this day of the week for the shift. **Note:** Used only when SMSHIFTYPE=D.
SMTHURSDAY	Alphanumeric, length=1	A flag that indicates whether this day of the week is used for this shift. 0=Do not use this day of the week for the shift. 1=Use this day of the week for the shift. **Note:** Used only when SMSHIFTYPE=D.

Column	Value	Description
SMFRIDAY	Alphanumeric, length=1	A flag that indicates whether this day of the week is used for this shift. 0= Do not use this day of the week for the shift. 1= Use this day of the week for the shift. **Note:** Used only when SMSHIFTYPE=D.
SMSATURDAY	Alphanumeric, length=1	A flag that indicates whether this day of the week is used for this shift. 0=Do not use this day of the week for the shift. 1=Use this day of the week for the shift. **Note:** Used only when SMSHIFTYPE=D.
SMSUNDAY	Alphanumeric, length=1	A flag that indicates whether this day of the week is used for this shift. 0=Do not use this day of the week for the shift. 1=Use this day of the week for the shift. **Note:** Used only when SMSHIFTYPE=D.
SMSHIFTYPE	Alphanumeric, length=1	The flag indicating whether the shift date range or shift day of the week is used. S=By date range. D=By day of week.
SMROWS	Integer	The shift's row threshold.
SMTIME	Integer	The shift's time threshold.
SMIOS	Integer	The shift's IOS threshold.
SMSTATUS	Integer	The status of the rule induction process. 0=Complete 1=Submitted 2=In progress
SMKNBTYPE	Alphanumeric, length=1	A flag that indicates the Knowledge Base type. D=Data rules C=Custom rules B=Both

This database is updated during the Build Rules process or the Custom Rule Builder wizard and contains all PRL statements generated by the data induction process or the custom rule build wizard.

Note: This database is created during installation and configuration. It is only used with Resource Governor.

Column	Value	Description
SMKNBNAME	Alphanumeric, length=8	The rule file name used for governing.
SMSEGNUM	Integer	The sequence number.
SMPRLLINE	Alphanumeric, length=80	Each line of the rule file governing program written in Production Rule Language (PRL).
SMPRLTYPE	Alphanumeric, length=1	The Production Rule Language (PRL) code was produced by the WebFOCUS Resource Analyzer rule engine or by custom rule building. A=Produced by rule engine. C=Produced by custom rule building.

SMPARAMETERS Database (SMPRMTRS.MAS)

This database contains all shift parameters and thresholds.

Note: This database is created during installation and configuration. It is only used with Resource Governor.

Column	Value	Description
SMSHIFTNAME	Alphanumeric, length=8	The shift name.
SMSTARTTIME	Alphanumeric, length=4	The start time for the thresholds to take effect. The format is '*hhmm*,' which designates the hours and minutes.
SMENDTIME	Alphanumeric, length=4	The end time for the thresholds to take effect. The format is '*hhmm*,' which designates the hours and minutes.
SMSTARTDATE	Alphanumeric, length=4	The start date for the thresholds to take effect. The format is '*MMDD*,' which designates the month and day.

Column	Value	Description
SMENDDATE	Alphanumeric, length=4	The end date for the thresholds to take effect. The format is '*MMDD*,' which designates the month and day.
SMCPUTIME	Integer	The CPU time threshold is the CPU seconds that are allowed to be consumed by the server process executing a request.
SMELAPTIME	Integer	The elapsed time threshold is the physical seconds that are allowed to be consumed by the server process executing a request.
SMROWS	Integer	The threshold representing the maximum number of result rows that can be returned to a client.
SMIOS	Integer	The threshold representing the maximum number of IOs that are allowed to be consumed by the server process executing the request.
SMUSED	Alphanumeric, length=1	The status flag indicating that the shift is in use. 0= The shift is not in effect for all subsequent rule building. 1= The shift is in effect for all subsequent rule building.
SMMONDAY	Alphanumeric, length=1	A flag that indicates whether this day of the week is used for this shift. 0=Do not use this day of the week for the shift. 1=Use this day of the week for the shift. **Note:** Used only when SMSHIFTYPE=D.
SMTUESDAY	Alphanumeric, length=1	A flag that indicates whether this day of the week is used for this shift. 0=Do not use this day of the week for the shift. 1=Use this day of the week for the shift. **Note:** Used only when SMSHIFTYPE=D.

Column	Value	Description
SMWEDNESDAY	Alphanumeric, length=1	A flag that indicates whether this day of the week is used for this shift. 0=Do not use this day of the week for the shift. 1=Use this day of the week for the shift. **Note:** Used only when SMSHIFTYPE=D.
SMTHURSDAY	Alphanumeric, length=1	A flag that indicates whether this day of the week is used for this shift. 0=Do not use this day of the week for the shift. 1=Use this day of the week for the shift. **Note:** Used only when SMSHIFTYPE=D.
SMFRIDAY	Alphanumeric, length=1	A flag that indicates whether this day of the week is used for this shift. 0= Do not use this day of the week for the shift. 1=Use this day of the week for the shift. **Note:** Used only when SMSHIFTYPE=D.
SMSATURDAY	Alphanumeric, length=1	A flag that indicates whether this day of the week is used for this shift. 0= Do not use this day of the week for the shift. 1=Use this day of the week for the shift. **Note:** Used only when SMSHIFTYPE=D.
SMSUNDAY	Alphanumeric, length=1	A flag that indicates whether this day of the week is used for this shift. 0=Do not use this day of the week for the shift. 1=Use this day of the week for the shift. **Note:** Used only when SMSHIFTYPE=D.
SMSHIFTTYPE	Alphanumeric, length=1	The flag indicating whether the shift date range or shift day of the week is used. S=By date range. D=By day of week.

Usage Monitoring Databases

This section lists the data definitions that comprise the Resource Analyzer Usage Monitoring Databases, and provides an explanation of the column values. A unique index is also created for the SMQUERY database.

SMQUERY Database (SMQUERY.MAS)

This contains the analysis of the query. This the main record created for each request that is monitored or governed.

Column	Value	Description
SMKEY	Alphanumeric, length=40	The request key.
SMDATE	Alphanumeric, length=8	The date in the format '*YYYYMMDD*'.
SMTIME	Alphanumeric, length=6	The time in the format '*hhmmss*'.
SMUSERID	Alphanumeric, length=30	The user identifier that made the connection to the server.
SMGROUPNAME	Alphanumeric, length=8	The security group identifier used for the current user ID.
SMEDASERVER	Alphanumeric, length=8	The server name as defined in the GKESET FOCEXEC file.
SMEDASERVTYPE	Integer	The server type as defined in the GKESET FOCEXEC file. 1=Gateway 2=Full-Function 3=Hub
SMCOLLECT	Alphanumeric, length=1	0 = Usage Monitor is turned on in GKTABLE 1=Unable to estimate 2=Cancelled 3=ADVISE
SMALLROWS	Alphanumeric, length=1	0=Incomplete result set 1=Complete result set 2=Incomplete, resource limited by the data engine

Column	Value	Description
SMCORRQRY	Alphanumeric, length=1	Indicates whether this is a correlated request. 0=No 1=Yes
SMSETALL	Alphanumeric, length=1	Indicates whether SET ALL is used. 0=No 1=Yes
SMELAPTIME	Integer	The amount of elapsed time used by the request in seconds.
SMCPUTIME	Integer	The amount of CPU time used by the request in milliseconds.
SMROWS	Integer	The number of result rows returned to client.
SMIOS	Integer	The number of input/output operations used by the server to satisfy the request.
SMRECLIMIT	Integer	The maximum number of result rows that has been set by the request or by the server. The value is zero if there is no limit.
SMHOLDFILE	Alphanumeric, length=8	The name of the HOLD file used.
SMRPCNAME	Alphanumeric, length=66	The long remote procedure name if the request originated in a remote procedure.
SMUNIONS	Integer	The number of UNION statements in the request.
SMUNIONALLS	Integer	The number of UNION ALL statements in the request.
SMNUMFROMS	Integer	The number of data types used in the request. Also, the number of rows entered in SMFROMS database.
SMNUMCOLUMNS	Integer	The number of columns used in the request. An asterisk (*) will be counted as 1.

Column	Value	Description
SMNUMREQUESTS	Integer	The number of rows stored in the SMREQUESTS database, based on 72 characters per line of the original request.
SMNUMRELATIONS	Integer	The number of relational clauses in the request.
SMNUMGROUPBYS	Integer	The number of GROUP BYs in the request.
SMNUMORDERBYS	Integer	The number of ORDER BYs in the request.
SMNUMBYS	Integer	The number of FOCUS BYs in the request.
SMNUMFUNCTIONS	Integer	The number of functions used in the request.
SMREADLIMIT	Integer	The read limit value.
SMVIEWSEG	Integer	The view segment number.
SMVIEWFLD	Integer	The view field number.
SMVIEWIDX	Integer	The view index number, or 0 for none.
SMCACHESIZE	Integer	The size of cache in 4K pages.
SMQUERYTYPE	Integer	The query type: 0=SQL 1=Table 2=TableF 8=Modify 25=Graph 28=Analyse 32=Match 66=Maintain
SMHYPERFOC	Alphanumeric, length=1	Reserved.
SMEXTSORT	Alphanumeric, length=1	Indicates whether external sort was used. 0=No 1=Yes

Column	Value	Description
SMLIVE	Alphanumeric, length=1	Indicates whether request is interactive, batch, or network. T=Interactive N=Network S=Batch
SMOUTPUT	Alphanumeric, length=1	Indicates whether output is online or offline. 0=Online 1=Offline
SMSCREEN	Alphanumeric, length=1	Indicates whether SET SCREEN is used. 0=No 1=Yes
SMSUNAME	Alphanumeric, length=8	The name used in the USE FOCUS command.
SMRECTYPE	Alphanumeric, length=1	Indicates the request type: S=SQL Select E=Execute C=Create (SQL) D=Drop (SQL) R=Alter (SQL) U=Update (SQL) I=Insert Into (SQL) D=Delete (SQL) T=Table (FOCUS) F=TableF (FOCUS) M=Match File (FOCUS) G=Graph (FOCUS) Y=Modify (FOCUS) N=Maintain
SMCONNADDR	Alphanumeric, length=32	Indicates the connection address: TCPIP SNA IPX MAC

Column	Value	Description
SMCONNTYPE	Integer	Indicates the connection type: 1=TCP 2=SNA 3=IPX
SMRPCLNO	Integer	The RPC line number of the request stack run.
SMROWWIDTH	Integer	The total byte count of row retrieval.
SMRPCKEY	Alphanumeric, length=40	The key of a single RPC.
SMRPCNUM	Integer	The level of an RPC call.
SMHOLDFORMAT	Integer	The format type indicator. (See SAHOLDFORMAT.)
SMHOLDTYPE	Integer	The hold type indicator. (See SAHOLDTYPE.)
SMREMARKS	Alphanumeric, length=20	The comment available for the administrator's use.
SABANDWIDTH	Integer	The computed value of SMROWS * SMROWWIDTH.
SAMO	Alphanumeric, length=2	The numeric value of the month of the request.
SAYEAR	Alphanumeric, length=4	The numeric value of the year of the request.
SAMONTH	Alphanumeric, length=9	The character value of the month of the request.
SACONNTYPE	Alphanumeric, length=7	Indicates the connection method used when the request was executed. 1=TCPIP 2=SNA 3=PX 0=UNKNOWN

Column	Value	Description
SAHOLDFORMAT	Alphanumeric, length=8	Indicates type of hold file being created: 1=DIF 2= ALPHA 3=LOTUS 4= WP 5=IFPS 6=SLYK 7=CALC 8=FOCUS 9=DOC 10=BINARY 11=POSTSC 12=PS 13=ROLLUP 14=HTML 15=HTMTAB 16=COMMA 17=FUSION 18=GIF 19=BMP 20=WMF 21=EXCEL 22=GRAPH 23=SCREEN 24=FXF 25=VIEWER 26=CLIP 27= RTF 28= PDF 29=EXL4 30=EXL2K 31= NOTES 32=TABS 33=FXSL 34=INTERNAL 35=WK1 36=DBASE 37=DBASEIII 0=UNKNOWN

Column	Value	Description
SAHOLDTYPE	Alphanumeric, length=8	Indicates the kind of HOLD being performed: 1=HOLD 2=PCHOLD 3=SAVE 4=SAVB 0=UNKNOWN
SACPUTIME	Integer	The amount of CPU time used by the request in seconds.

SMREQUESTS Database (SMREQSTS.MAS)

This contains the actual query that was monitored.

Column	Value	Description
SMKEY	Alphanumeric, length=40	The request key.
SMRPCNUM	Integer	The RPC number.
SMSEGNUM	Integer	The sequence number.
SMSQLLINE	Alphanumeric, length=72	A line of the request.

SMFROMS Database (SMFROMS.MAS)

Contains the list of databases that are queried against. This is always created, as any request must have at least one data source.

Column	Value	Description
SMKEY	Alphanumeric, length=40	The request key.
SMSEGNUM	Integer	The sequence number.
SMNAME	Alphanumeric, length=66	The database name or data type name used in the request.
SMDBMS	Alphanumeric, length=8	The name of the server engine when using Master/Access Passthru.

Column	Value	Description
SMSUFFIX	Alphanumeric, length=8	The engine type of the source, for example, FOC, SQLMSS.
SMALIAS	Alphanumeric, length=66	The name defined by the USE command.
SMMORE	Alphanumeric, length=1	1=MORE FILE
SMUSE	Alphanumeric, length=1	1=via USE
SMSU	Alphanumeric, length=1	1=SU 0=ALLOCATED
SMSUNAME	Alphanumeric, length=8	The name of the FDS Server.

SMCOLUMNS Database (SMCOLMNS.MAS)

This contains information about the columns used in queries. A select * against a relational database using Passthru will not have an smcolumns table update.

Column	Value	Description
SMKEY	Alphanumeric, length=40	The request key.
SMSEGNUM	Integer	The sequence number.
SMNAME	Alphanumeric, length=66	The database name or data type name used in the request.
SMCOLUMN	Alphanumeric, length=66	The column or field name used in the request.
SMDISTROWS	Alphanumeric, length=1	Indicates whether DISTINCT was used for this column. 0=No 1=Yes

Column	Value	Description
SMALLCOLS	Alphanumeric, length=1	Indicates whether a SELECT * or FOCUS PRINT was used. 0=No 1=Yes
SMDEFINES	Alphanumeric, length=1	Indicates what type of temporary field is created (the default is relational). 0=Unknown 1=Real 2=Permanent define 3=Temporary define 4=Compute
SMLITERAL	Alphanumeric, length=32	The literal used for the column.
SMFORMAT	Alphanumeric, length=8	The format of the field or column.
SMUSAGE	Integer	The Master File usage of field or column. 0=Unknown 1=Integer 2=Float 3=Double 4=Pack 5=Zone 6=Alpha 7=Hex 8=Exponent 9=Date 11=Text 12=DBCS 13=BLOB 14=CLOB 15=Time 16=Datetime 23=JDE_PACK
SMSIZE	Integer	The length of column in bytes.

Column	Value	Description
SMDEC	Integer	The number of places to the right of the decimal.
SMIDXKEY	Alphanumeric, length=1	Is this an indexed column in a Master File? 0=No 1=Yes
SATYPE	Alphanumeric, length=16	Indicates the type of field: 1=Real field defined in the Master File 2=Field DEFINEd in the Master File 3=Field DEFINEd in the procedure 4=Field COMPUTEd in the procedure 0=Unknown
SAUSAGE	Alphanumeric, length=8	Indicates the format of the field: 1=INTEGER 2=FLOAT 3=DOUBLE 4=PACK 5=ZONE 6=ALPHA 7=HEX 8=EXPONENT 9=DATE 11=TEXT 12=DBCS 13=BLOB 14=CLOB 15=TIME 16=DATETIME 23=JDE_PACK 0=UNKNOWN

SMRELATIONS Database (SMRELTNS.MAS)

This tracks the operations used on the column. This is updated only if the request has a *where* or *if* test.

Column	Value	Description
SMKEY	Alphanumeric, length=40	The request key.
SMSEGNUM	Integer	The sequence number.
SMOPERATOR	Alphanumeric, length=8	The operator used in a clause, such as '<' or '='.
SMAOPERATOR	Alphanumeric, length=1	The arithmetic operator used in a clause, such as '+' or '*'.
SMANDOR	Alphanumeric, length=3	The connector in or between clauses.
SMRNAME	Alphanumeric, length=66	The database name or data type name used in the request.
SMRCOLUMN	Alphanumeric, length=66	The right column or field name used in the relation.
SMLNAME	Alphanumeric, length=66	The database name or data type name used in the request.
SMLCOLUMN	Alphanumeric, length=66	The left column or field name used in the relation.
SMNOT	Alphanumeric, length=1	Indicates whether the NOT keyword was used. 0=No 1=Yes
SMALL	Alphanumeric, length=1	Indicates whether the ALL keyword was used. 0=No 1=Yes
SMANY	Alphanumeric, length=1	Indicates whether the ANY keyword was used. 0=No 1=Yes

Column	Value	Description
SMLITERAL	Alphanumeric, length=32	The literal used.
SMEXISTS	Alphanumeric, length=1	Indicates whether the EXISTS keyword was used. 0=No 1=Yes
SMRELTYPE	Alphanumeric, length=8	Indicates whether a WHERE, HAVING, or IF was used in this relation clause.
SMCORRELATED	Alphanumeric, length=1	Indicates whether this relation caused a correlation. 0=No 1=Yes
SMCORRCOLUMN	Alphanumeric, length=1	Indicates whether a column used caused a correlation. 0=No 1=Yes

SMBYS Database

This contains the BY or ORDER BY column name or number.

Column	Value	Description
SMKEY	Alphanumeric, length=40	The request key.
SMSEGNUM	Integer	The sequence number.
SMNAME	Alphanumeric, length=66	The database name or data type name used in the request.
SMCOLUMN	Alphanumeric, length=66	The column or field name used as the target of the SORT BY or GROUP BY.
SMCOLNBR	Alphanumeric, length=4	The column number used in an SQL ORDER BY.
SMORDER	Alphanumeric, length=1	A=Ascending sort D=Descending sort
SMBYTYPE	Alphanumeric, length=1	Indicates whether sorting or grouping was performed on the column. B=Sort G=SQL Group

SMFUNCTIONS Database (SMFNCTNS.MAS)

Tracks functions used on columns. This is updated if the request used a function such as MIN, MAX, SUM, or AVE.

Column	Value	Description
SMKEY	Alphanumeric, length=40	The request key.
SMSEGNUM	Integer	The sequence number.
SMALLROWS	Alphanumeric, length=1	Indicates whether '*' was used.
SMFUNCTION	Alphanumeric, length=12	The function used.
SMNAME	Alphanumeric, length=66	The database name or data type name used in the request.
SMCOLUMN	Alphanumeric, length=66	The column or field name used.
SMLITERAL	Alphanumeric, length=32	The literal used.

SMGOVERN Database (SMGOVEND.MAS)

This database is updated if a request is governed or advise is on.

Note: This database is created during installation and configuration. It is only used with Resource Governor.

Column	Value	Description
SMKEY	Alphanumeric, length=40	The request key.
SMELAPTHRESH	Integer	The elapsed time threshold in effect.
SMROWTHRESH	Integer	The result row threshold in effect.
SMIOTHRESH	Integer	The IO threshold in effect.
SMCANCELTHRESH	Alphanumeric, length=8	The threshold that was exceeded by name. This can be ELAPSED, ROWS, NOEST, or RUN.
SMRULENUM	Integer	The rule number that caused the governing result or decision.
SMKBNAME	Alphanumeric, length=8	The rule files name that governed.

CHAPTER 6

Server Configuration File Keywords

Topic:

- MVS Server Configuration File Keywords

These topics provide details on keywords in the server configuration file.

MVS Server Configuration File Keywords

Each valid keyword associated with the server configuration file is described in the following topics. The following types of keywords are described:

- Global
- Service
- Servinit

Reference **Keywords and Service Blocks**

A server configuration file is made up of keywords and service blocks.

Keywords A keyword describes a unique attribute of a server.

```
KEYWORD=value
```

for example:

```
EXTSEC=ON
```

Service Blocks A server configuration file is divided into sections called service blocks. A service block defines one instance of a server type.

Global Keywords

Global keywords are described in the following table:

Keyword	Description
APFAUTH	Enables the system administrator, on an APF-authorized server, to provide a non APF-authorized load library allocation. INTERNAL Allows load libraries, which need to be allocated to the server but do not require APF-authorization, to be included in the server JCL. This keyword is used in conjunction with the TASKLIB keyword that is used to provide an alternate ddname.
EDASHARE	Allows a client application to interrupt a query. ON Activates the query interrupt feature. **Note:** If this feature is not required, do not code this parameter.

Keyword	Description
EXTSEC	Enables your site to activate external security and provide user-level security privileges and restrictions managed by external security packages, such as RACF, CA-ACF2, and CA-TOP SECRET. Possible values are: ON enables external security. The STEPLIB libraries in the server JCL must be APF-authorized for this setting. OFF disables external security. Connecting users will inherit the security privileges and restrictions of the user ID of the server address space.
FASTLOAD	Enables the server to automatically retain software modules in virtual memory. **AUTO** Enables this feature. **Note:** This keyword is included in the startup configuration file.
LICENSE	Is a 12 digit alphanumeric code that was supplied in your packaging. **Note:** This keyword is included in the startup configuration file.
LONGSYNM	Enables the server to support synonym names greater than 8 characters. ON Enables support. Synonym names can be from 1 to 64 characters in length. **Note:** This keyword is included in the startup configuration file.
MN_INTERVAL	Specifies the interval at which to perform monitoring (when active). *nn* This value, in seconds, is an integer indicating the interval. The minimum is 30 seconds and the maximum is 3,600 seconds. The default value is 1,800 seconds (30 minutes). You can activate monitoring to perform at integer multiples of the MRM_INTERVAL (MVS Resource Manager Interval). **Note:** APF authorization is required.

Keyword	Description
MN_REGION	Activates monitoring to include region statistics. Possible values are: ON specifies monitoring to include region statistics. OFF specifies monitoring not to include region statistics. The default value is OFF. **Note:** APF authorization is required.
MN_STORAGE	Activates monitoring to include storage statistics. Possible values are: ON specifies monitoring to include storage statistics. OFF specifies monitoring not to include storage statistics. The default value is OFF. **Note:** APF authorization is required.
MN_USERS	Monitors statistics by user ID. Possible values are: OFF limits monitoring of user IDs. The default value is OFF. ALL produces usage statistics for all logged on user IDs. **Note:** APF authorization is required.
MONITOR	Provides detailed MVS resource utilization reports for a region or for all users. Possible values are: ON enables monitoring on this server. OFF disables monitoring. This setting will reset the current settings for MN_USER and MN_INTERVAL to the original values in the configuration file. The default value is OFF. **Note:** APF authorization is required.

Keyword	Description
MRM	Specifies the status (active or inactive) of the Resource Manager. The Resource Manager is used by the server to ensure that system resources are allocated appropriately among all of the tasks in the server address space. In addition to balancing the CPU allocated to each task, the Resource Manager allows assignment and control of the relative priorities of the tasks in the region. When the Resource Manager is active, the SSTIME task wakes up every 10 seconds or the value specified for MRM_INTERVAL (see the next keyword entry), and analyzes the tasks in the region. The MVS dispatcher, once it has selected the server address space, scans the TCB chain until it finds the first ready TCB, and gives it control. Consequently, tasks early in the TCB chain get more access to the CPU, and a CPU-bound (looping) task can prevent other tasks from running. Within each Resource Manager priority group, tasks compete with each other for CPU time. The Resource Manager looks at the tasks within a priority group, and if the top task is using a percentage of the CPU exceeding 50%, or the MRM_PERCENT (see the keyword entry) value, the Resource Manager changes the task's MVS priority to place it at the end of the group. In addition, totally inactive tasks, which consume no CPU and are in a wait state, are given a very low background priority. This feature helps the MVS dispatcher by making the chain shorter for active users, and therefore getting them dispatched faster. When a background task becomes active again, it is reassigned its original priority. Possible values are: ON activates the Resource Manager. This is the default value if running with APF authorization and the keyword SMFNUM has been specified or allowed to default to 0. OFF deactivates the Resource Manager. Use this setting only if you suspect a problem with the Resource Manager.
MRM_INTERVAL	Specifies the second interval in which the Resource Manager will monitor all users to prevent any users from exceeding the MRM_PERCENT value (see the next keyword entry). Valid values are from 1 to 1,400. The default is 10. **Note:** APF authorization is required.

Keyword	Description
MRM_PERCENT	Specifies the threshold percentage of CPU that will cause a user to be reduced in priority by the Resource Manager. Valid values are from 1 to 99. The default is 50. **Note:** APF authorization is required.
RACSP	Specifies the MVS storage subpool used when SAF creates its security control blocks. This keyword enables definition of the subpool area in which the ACEEs are created for UID and their security environment. Valid values for RACSP are 230 or 250. The default value is 230.
SMFNUM	Enables user-level usage accounting for a server. SMFNUM is set representing a user SMF record number, from 128 to 255, inclusive. SMFNUM=0 is allowable, and has the effect of enabling internal usage accounting (for operator command displays, and for the console feature) without generating SMF records. **Note:** APF authorization is required.
SNAPEXCLUDE	Allows you to selectively exclude the IBISNAP for selected abends. SNAPEXCLUDE=*code,reason* where: *code* Is the abend code. An * is allowed for the last character to indicate a wildcard. *reason* Is the reason code for the abend. A value of * indicates all reason codes. If * is not used, then a reason code must be given. The use of the * as a wildcard as part of the reason code IE 1*, is not allowed. For example, SNAPEXCLUDE=S913,38 SNAPEXCLUDE=S91*,* **Note:** SOCx codes can also be excluded.

Keyword	Description
SOS_PERCENT	Controls Short on Storage (SOS) monitoring. Specifies the percentage of storage utilization, which when exceeded, causes a warning message to be issued using non-scrollable messages to the Server Console and EDAPRINT. You can alter the message destination with the Enhanced Message Routing Facility. If storage utilization falls below this value, a message is issued stating that the storage constraint is relieved and the WTO message is deleted (unless suppressed by the Enhanced Message Routing Facility). The SOS_PERCENT check, if active, is made at every MVS Resource Manager interval (MRM_INTERVAL). SOS_PERCENT enables SOS monitoring when it is an integer representing a percentage greater than 0 up to 99 percent. A value of 0 disables SOS monitoring. The default value is 0. **Note:** APF authorization is required.
STORAGEABOVE	Is a decimal value indicating how many kilobytes are needed above the line to start additional subtasks within the server. If less than the amount specified is available, no additional logons are permitted. The default value is 4,096 (for 4,096K).
STORAGEBELOW	Is a decimal value indicating how many kilobytes are needed below the line to start another subtask within the server. If less than the amount specified is available, no additional logons are permitted. The default value is 512 (for 512K).

Keyword	Description
SZERO	Controls the use of memory subpool zero in the server. Possible values are: NO causes subpool 0 to not be shared between subtasks. This eliminates the possibility that any programs or services invoked within the server may leave unfreed storage behind (the server itself uses subpool 15 for general memory needs). YES should be used when you include VSAM data sets in your server startup JCL (we also refer to these allocations as global ddnames). The VSAM access method assumes subpool zero is shared, and shares certain control blocks between tasks when the ddname is shared between them. This can cause a S0C4 if subpool zero is in fact NOT shared. If a VSAM data set is allocated to an individual subtask using DYNAM, then the ddname is not shared and this problem will not arise.
TASKLIB	Enables the System Administrator to reduce the APF authorization requirements of a server. Specify TASKLIB=ddname for data sets that need to be allocated to the server but do not require APF authorization, where ddname refers to a ddname allocation in the server startup JCL. When looking for a specified program to execute, the server searches ddname, which points to programs that are not APF authorized. If the server does not find the program, it then searches STEPLIB, which points to programs that have APF authorization. **Note:** • The data set *qualif*.EDALIB.LOAD must be allocated to STEPLIB. • This keyword requires APFAUTH=INTERNAL.
TRUST_NT	Y Allows an NT server or client to connect as a trusted platform. N Does not allow an NT server and/or client to be trusted. This value is the default.

Keyword	Description
`UNIQUE`	Restricts the number of users that can access a server. UNIQUE=logid prevents more than one user from logging on with the given logid. UNIQUE=secid prevents more than one user from logging on with the given secid. When this keyword is omitted in the server startup configuration file, the server will allow any number of users to log on with the same logid or secid.
`WLM_ENCLAVE_TRNAME`	`WLM_ENCLAVE_TRNAME = TRANSACTION NAME or Service Class` Allows an enclave name to be specified. This enclave is a feature of Workload Manager (WLM).

Service Keywords

Service keywords are described in the following table:

Keyword	Description
CPU_LIMIT	Is the total amount of CPU time, in seconds, that a single connected user can consume before the Resource Manager terminates the user task.
DEPLOYMENT	Indicates the deployment mode. PRIVATE (default) Each user of the server will get a private task, which will be cleared when they disconnect so the task will be re-used by the next user. NOTE: PDS indexes will not be re-read in this mode. This is not true for server profiles, which are always refreshed for each user connection. PRIVATE,REREAD Same as for PRIVATE but all PDS files will re-read for each user. Should only be used for EDADBA service or when in development mode. ISOLATED Each user of the server will get a private task, which will be deleted when they disconnect from the server. POOLED Each user of the server will re-use a task and context of last user of the task. Task will not be deleted on user disconnect.
IDLELIM	Specifies the number of seconds that the server will allow a given subtask or the server region to remain idle before termination. This feature automatically enables the server to terminate a subtask or the server region if either of them remains idle for the specified period of time.

Keyword	Description
MAXIMUM	Specifies the maximum number of agents allowed at the same time. This value defines the number of users allowed for the service. The total number on a single server cannot exceed 394. **Note:** This keyword is included in the system-supplied service block.
MEMORY_LIMIT_ABOVE	Is the maximum amount of above-the-line storage, in kilobytes, that a single user agent can allocate before the user task is cancelled.
MEMORY_LIMIT_BELOW	Is the maximum amount of below-the-line storage, in kilobytes, that a single user agent can allocate before the user task is cancelled.
PROGRAM	Specifies the server program to be run. Possible values are: TSCOM3 applies to all communications protocols except LU2. TSCOM3L2 applies to LU2 only. **Note:** This keyword is included in the system-supplied service block.
PRTYGROUP	Defines the Resource Manager priority of a given service, from 0 to 15. 0 is the lowest priority and 15 is the highest. If 0 is specified, user activity will be suspended until a higher priority is specified. The default value is 7.
REFRESH_LIMIT	Is the parameter for DEPLOYMENT=POOLED or DEPLOYMENT=PRIVATE only. It specifies the number of connect sessions to allow before the application agent task is refreshed. The default is the application agent is not refreshed until the server is restarted.

Keyword	Description
SERVICE	Identifies, and names, a server service. Names should be unique, and from 1 to 8 characters in length. The keywords and values that follow the SERVICE specification in a service block define the type of service to be provided. **Note:** This keyword appears in the system-supplied service block.
TASK_RESTART	Restarts all private application agent tasks within a server by default, after the client disconnects to enhance the performance of reconnects. Possible values are: ON the application agent restarts after task termination. OFF the application agent will not restart until a new user connection requires it.

SERVINIT Keywords

Servinit keywords are described in the following table:

Keyword	Description
SERVINIT	Defines the Service Initialization File to be invoked for a service. This value identifies a member of the partitioned data set allocated to the ddname EDACOM in the server startup JCL. It contains the commands to be executed in order to set up an agent that is to run the service. It can be specified instream by the value *,++, where ++ is the termination string for the file. It is recommended that this file be created instream. **Note:** This keyword appears in the system-supplied service block.
USERID	Is optional with pooled deployment. It must be used in conjunction with the PASSWORD parameter. If not used, a POOLED user will inherit the security profile of the user who submitted the server for MVS.
PASSWORD	Is optional with pooled deployment. Must be used in conjunction with the USERID parameter.
APPROOT	Specifies the high level qualifier to be used for the Application Name Space data sets.

CHAPTER 7

MVS Accounting

Topics:

- Server Accounting
- Setting the Accounting Field
- Accessing Accounting Statistics
- SMF Record Format

The server provides an optional facility (SMFNUM) to use for accounting purposes that enables you to log resource utilization on a per-user basis. This facility enables the server to generate SMF records for query-level and user-level accounting.

Server Accounting

Server accounting requires that the server STEPLIB data sets be APF-authorized. When SMFNUM is set to a non-zero value, an SMF record is generated, containing the:

- Logon ID and security ID of the user.
- CPU time and EXCPs consumed.
- Data based on the type of record written.

You can process the SMF records using the accounting programs that exist at your site. Examples of SMF records are provided in *SMF Record Format* on page 7-4.

Syntax How to Enable Accounting

To enable accounting, insert the following statement into the server configuration file:

```
SMFNUM=smfnumber
```

where:

smfnumber

Is an integer value equal to 0 or in a range from 128 to 255, inclusive. This number represents the SMF number used by the accounting facility when it sends records to the SMF system. If the SMF number is set to 0, then no SMF records are created, but statistics are maintained by the server for internal monitoring.

Setting the Accounting Field

Up to 40 characters can be supplied that appear in the SMF Records field SMFOFA40. The following command can be used in any support server profile to provide the account field information.

```
SET BILLCODE=value
```

where:

value

Is the 1– 40 characters to be used on each SMF record produced. This information can also be set dynamically from a client application by coding an RPC with the SET command and executing it with the value as a parameter. WebFOCUS users can send the SET command to the server.

Accessing Accounting Statistics

Master Files are provided for accessing accounting statistics, as members SMFVSAM and SMFFIX of the EDAMFD.DATA data set. These Master Files enable you to interpret the SMF records generated by the accounting facility using stored procedures.

To process these records and interpret the statistics, you can issue a stored procedure using SQL and the Dialogue Manager language to generate any report your site may require.

Note:

- Before you query the SMFVSAM or SMFFIX databases, ensure that the SMF VSAM file or the SMF DUMP file is allocated, using the DYNAM command, to a non-system PDS.
- The SMF Master Files provided are for logoff records only. This is indicated by ALIAS=2. Currently, there are five RECTYPE values defined to produce SMF records, as follows:

RECTYPE:

Value	Description
1	Indicates a start of task record. When included in a report, these statistics tell when a task initiation occurred, and are of no particular use in chargeback. By pairing start and end of task records for all tasks within a time period, statistics such as average active time, peak task count, and average task count can be determined. These values can be used for future capacity planning activities for the server.
2	Indicates the start of a task record. When included in a report, these statistics tell when a task termination occurred. These records are cut for both publicly and privately deployed services and contain statistics for the subtask as a whole. For privately deployed services, RECTYPE (2) records contain statistics associated with a single user connection.
4	Begin query. (Record layout is the same as RECTYPE (1).
5	End query. (Record layout is the same as RECTYPE (2).

Accounting for DB2 in a Server Task

When using a server to access DB2 data, certain processing takes place within the DB2 address space and is governed by DB2's chargeback system. If a user requests data from DB2, the server passes the request to the DB2 subsystem. The DB2 subsystem then processes the request, performing such tasks as retrieving rows and aggregating the data. It generates the answer set, and passes the output back to the server. The server then performs any joins and formatting which have not been performed by DB2 to satisfy the original request.

Charges incurred while the request was being processed by the DB2 subsystem are added to the charges accumulated in the server task that originated the request for processing. If the server accounting is enabled, these charges are associated with the user's logon and security IDs in the SMF records described earlier.

SMF Record Format

The record formats for the SMF records written by the server are defined below. The formats are provided in the system 390 assembler DSECT form.

RECTYPES 1 and 4

```
SMFON    DSECT
         SPACE
*-----------------------------------------------------*
*  USAGE ACCOUNTING SMF RECORD LAYOUT FOR LOGON RECORDS.              *
*                                                                     *
*  THIS IS THE DSECT DESCRIBING THE SMF RECORD WHICH IS PASSED TO     *
*  YOUR EXIT ON AT USER LOGON TIME.  IT IS COMPLETELY READY TO BE     *
*  WRITTEN WHEN YOUR EXIT RECEIVES CONTROL.                           *
*-----------------------------------------------------*
         SPACE
*-----------------------------------------------------*
*  THE FIRST TWENTY FOUR BYTES OF THE RECORD ARE THE SMF HEADER.      *
*  THESE FIELDS ARE REQUIRED IN ALL SMF RECORDS (18 BYTES FOR RECORDS *
*  WITHOUT SUBTYPES; WE USE SUBTYPES, THE HEADER IS 24 BYTES).        *
         SPACE
SMFONLEN DS    H'116'            RECORD LENGTH
SMFONSEG DS    XL2'0000'         SEGMENT DESCRIPTOR (0 UNLESS SPANNED)
SMFONFLG DS    XL1               SYSTEM INDICATOR
SMFONRTY DS    XL1               RECORD TYPE
SMFONTME DS    XL4               TIME, IN HUNDREDTHS OF A SECOND
SMFONDTE DS    PL4               DATE, 00CYYDDDF, WHERE F IS THE SIGN
SMFONSID DS    CL4               SYSTEM IDENTIFICATION
SMFONSBS DS    CL4               SUBSYSTEM IDENTIFICATION
SMFONSBT DS    XL2'0001'         SUBTYPE OF RECORD - X'0001' INDICATES X
                                 THIS IS A LOGON RECORD
         SPACE
```

```
*-----------------------------------------------------*
*  THE NEXT FIELDS ARE THOSE PRESENT IN THE LOGON                  *
*  RECORD FOR THE START OF A USER SESSION.                         *
*-----------------------------------------------------*
         SPACE
SMFONMSO DS    CL8              JOBNAME
SMFONJID DS    CL8              JOBID (FROM SSIBJBID)
SMFONASI DS    Y                ASID
SMFONRV1 DS    XL2              RESERVED
SMFONUID DS    CL8              SECURITY USERID
SMFONLID DS    CL8              USERID PRESENTED AT LOGON (SAME AS    X
                                SMFONSID UNLESS CHANGED VIA MSIDTR    X
                                SECURITY EXIT)
SMFONRUL DS    CL8              TSO USERID/CICS REGION/LU NAME        X
                                CONTENTS OF THIS FIELD IS TSO USERID  X
                                IF SMFONCNT = SMFONTSO, CICS REGION   X
                                (JOBNAME) IF SMFONCNT = SMFONCIC,     X
                                OR LU NAME IF SMFONCNT = SMFONVTM
SMFONCTI DS    CL4              WHEN SMFONCNT = SMFONCIC, THIS FIELD  X
                                CONTAINS THE CICS TERMID
SMFONSRV DS    CL8              SERVICE NAME FROM SERVICE BLOCK
SMFONRS0 DS    XL4              RESERVED FOR FUTURE EXPANSION
SMFONCNT DS    XL1              CONNECTION TYPE
         SPACE
SMFONTSO EQU   1                CONNECTION VIA TSO
SMFONCIC EQU   2                CONNECTION VIA CICS
SMFONVTM EQU   4                CONNECTION VIA VTAM
SMFONPSR EQU   8
         SPACE
SMFONRS1 DS    XL3              RESERVED
SMFONID1 DS    F                SYSPLEX ID 1
SMFONID2 DS    F                SYSPLEX ID 2
SMFOFPID DS    XL8              POOLED USER ID
SMFONRS2 DS    XL12             RESERVED
SMFONL   EQU   *-SMFON          LENGTH OF THE SMF LOGON RECORD
```

RECTYPES 2 and 5

```
SMFOF    DSECT
         SPACE
*-------------------------------------------------*
*  USAGE ACCOUNTING SMF RECORD LAYOUT FOR LOGOFF RECORDS.          *
*                                                                 *
*  THIS IS THE DSECT DESCRIBING THE SMF RECORD WHICH IS PASSED TO  *
*  YOUR EXIT ON AT USER LOGOFF TIME.  IT IS COMPLETELY READY TO BE *
*  WRITTEN WHEN YOUR EXIT RECEIVES CONTROL.                        *
*-------------------------------------------------*
         SPACE
*-------------------------------------------------*
*  THE FIRST TWENTY FOUR BYTES OF THE RECORD ARE THE SMF HEADER.   *
*  THESE FIELDS ARE REQUIRED IN ALL SMF RECORDS (18 BYTES FOR RECORDS *
*  WITHOUT SUBTYPES; WE USE SUBTYPES, THE HEADER IS 24 BYTES).     *
*-------------------------------------------------*
         SPACE
SMFOFLEN DS    H'168'           RECORD LENGTH
SMFOFSEG DS    XL2'0000'        SEGMENT DESCRIPTOR (0 UNLESS SPANNED)
SMFOFFLG DS    XL1              SYSTEM INDICATOR
SMFOFRTY DS    XL1              RECORD TYPE
SMFOFTME DS    XL4              TIME, IN HUNDREDTHS OF A SECOND
SMFOFDTE DS    PL4              DATE, 00CYDDDF, WHERE F IS THE SIGN
SMFOFSID DS    CL4              SYSTEM IDENTIFICATION
SMFOFSBS DS    CL4              SUBSYSTEM IDENTIFICATION
SMFOFSBT DS    XL2'0002'        SUBTYPE OF RECORD - X'0002' INDICATES X
                                THIS IS A LOGOFF RECORD
         SPACE
```

```
*-----------------------------------------------------*
*  THE NEXT FIELDS ARE THOSE PRESENT IN THE LOGOFF                  *
*  RECORD FOR THE END OF A USER SESSION.                            *
*-----------------------------------------------------*
         SPACE
SMFOFMSO DS    CL8             JOBNAME
SMFOFJID DS    CL8             JOBID (FROM SSIBJBID)
SMFOFASI DS    Y               ASID
SMFOFRV1 DS    XL2             RESERVED
SMFOFUID DS    CL8             SECURITY USERID
SMFOFLID DS    CL8             USERID PRESENTED AT LOGON (SAME AS    X
                               SMFOFSID UNLESS CHANGED VIA MSIDTR    X
                               SECURITY EXIT)
SMFOFRUL DS    CL8             TSO USERID/CICS REGION/LU NAME        X
                               CONTENTS OF THIS FIELD IS TSO USERID  X
                               IF SMFOFCNT = SMFOFTSO, CICS REGION   X
                               (JOBNAME) IF SMFOFCNT = SMFOFCIC,     X
                               OR LU NAME IF SMFOFCNT = SMFOFVTM
SMFOFCTI DS    CL4             WHEN SMFOFCNT = SMFOFCIC, THIS FIELD  X
                               CONTAINS THE CICS TERMID
SMFOFSRV DS    CL8             SERVICE NAME FROM THE SERVICE BLOCK
SMFOFRS0 DS    XL4             RESERVED FOR FUTURE EXPANSION
SMFOFCNT DS    XL1             CONNECTION TYPE
         SPACE
SMFOFTSO EQU   1               CONNECTION VIA TSO
SMFOFCIC EQU   2               CONNECTION VIA CICS
SMFOFVTM EQU   4               CONNECTION VIA VTAM
SMFOFPSR EQU   8
SMFOFCC  DS    XL3             COMPLETION CODE FOR THE TASK
SMFOFACT DS    CL8             USER ACCOUNTING INFORMATION; THIS     X
                               FIELD CURRENTLY PASSED AS LOW VALUE
SMFOFCPU DS    XL4             CPU TIME IN HUNDREDTHS OF A SECOND
SMFOFEXC DS    XL4             COUNT OF EXCP'S
SMFOFLTM DS    FL4             LOGON DURATION IN HUNDREDTHS OF A     X
                               SECOND
SMFPRTY  DS    XL1             PRIORITY
SMFCOMPL DS    XL1             COMPLETION TYPE
         DS    XL2             RESERVED
SMFOFID1 DS    F               SYSPLEX ID 1
SMFOFID2 DS    F               SYSPLEX ID 2
SMFOPID  DS    XL8             POOLED USERID
SMFOFA40 DS    CL40            FULL 40-BYTE ACCOUNTING FIELD
         SPACE
SMFOFL   EQU   *-SMFOF         LENGTH OF THE SMF LOGOFF RECORD
```

CHAPTER 8

Application Namespaces

Topics:

- Location of Application Components
- Application Namespace Maintenance Commands
- Application Namespace Run-Time CommandsApplication Namespace Run-Time Commands

These topics provides details on Application Namespaces. This feature allows all the components of an application to be held in a predefined area on an application-by-application basis.

Location of Application Components

Applications consist of a number of components, which are stored on the server. By default, the storage location is dependent on the component type. Application Namespaces, when active on a server, allow all the components of an application to be held in a predefined area on an application-by-application basis. In addition, it enables sharing of application components in an organized manner. This applies to static (for example, FOCEXECs, Masters, HTML, GIFs), temporary (components that are deleted after client disconnect), and external (data sources) components.

If the Application Namespace feature is not active, the behavior is the same as in earlier versions of the server. In earlier versions for UNIX, Windows NT/2000, OpenVMS, and OS/400, components are stored, by default, in the catalog directory under EDACONF and/or controlled by various variables (for example, EDAPATH, EDASYN). On MVS, the behavior is the same except that ddnames control where components are found.

The behavior makes it difficult to deploy a PC-developed application to a server platform. Using the Application Namespace feature, and platform independent commands, a PC-developed application deployment is simplified.

Syntax How to Configure for Application Namespaces

A configuration parameter called approot is used to define the base physical location of the application directories on UNIX, Windows NT/2000, OpenVMS, and OS/400, and the high-level qualifiers for application Data Set Names (DSNs) on MVS.

On non-MVS platforms, the approot is coded in the edaserve.cfg file. The syntax is:

```
approot=directory_path_name
```

For MVS, the parameter is coded in the servinit section of a service block in the EDASERVE ddname. The syntax is:

```
APPROOT=qualif1.qualif2…
```

If approot is removed from the servers configuration file, APP commands are disabled and an appropriate message is issued if an APP command is attempted.

Reference APP Commands

There are several commands (mostly with platform independent syntax) that control the usage of the Application Namespace. These commands have been divided into maintenance commands used to create and manage the Application Namespace Server environment and run-time commands.

Maintenance Commands	Description
`APP CREATE appname1 appname2 appname3 [-] [appname4 ...]`	Creates one or more application areas below the approot location. The appname can be a 1 to 68 character name. **Note:** On MVS, only one appname can be used and appname cannot exceed eight characters.
`APP DELETE appname1 appname2 appname3 [-] [appname4 ...]`	Deletes one or more application areas. The application area does not have to be empty. **Note:** Only one appname can be used on MVS.
`APP COPY appname1 appname2`	Copies a complete application from one application area to another.
`APP COPYF appname1 filename1 filetype1 appname2 filename2 filetype2`	Copies a single component from one application to another.
`APP MOVEF appname1 filename1 filetype1 appname2 filename2 filetype2`	Moves a single component from one application to another.
`APP RENAME appname1 appname2`	Renames a single application to appname2.
`APP RENAMEF appname filename1 filename2 filetype`	Renames a single component of a single type in appname to filename2.
`APP DELETEF appname filename filetype`	Deletes a single component from appname.

Maintenance Commands	Description
`APP LIST [HOLD]`	Lists all applications below approot. It is not affected by the APP PATH setting. If it is used with the HOLD option, output is written to focappl.ftm in the current temporary area, which can then be used in a report request.
`APP QUERY appname1 appname2 appname3 [-] [appname4 ...] [HOLD]`	Lists all components in one or more applications. If it is used with the HOLD option, output is written to focappq.ftm in the current temporary area, which can then be used in a report request. **Note:** Only one appname can be used on MVS.
`APP SHOWPATH`	Lists the application names in the currently active APP Path. Baseapp will always be listed last.
`APP HELP`	Provides help information on all commands.

Run-time Commands	Description
`APP ENABLE`	Enables the Application Namespace feature for the server.
`APP PATH appname1 appname2 appname3 [-] [appname4 ...]`	Sets the search path for application components.
`APP PREPENDPATH appname1 appname2 [-] appname3 [appname4 ...]`	Adds applications to the beginning of an existing APP PATH command.
`APP APPENDPATH appname1 appname2 [-] appname3 [appname4 ...]`	Adds applications to the end of an existing APP PATH command.
`APP HOLD appname`	Controls where all output files are created. This includes: `CREATE SYNONYM ON TABLE HOLD` (including SAVE) This command should be used with caution.
`APP FI filename DISK appname/physical name`	Provides platform independent syntax to allocate a file. For MVS, the file must already exist.
`APP MAP virtualname realdirectory`	Provides a mapping mechanism for application components outside of the APPROOT scope.

Application Namespace Maintenance Commands

The following maintenance commands are used to create an Application Namespace server environment. It is recommended that you place the run-time APP commands in the server's global profile. You can issue any of the maintenance commands using the EDAEXEC RPC. You can execute the EDAEXEC RPC from any client platform that can execute RPCs on the server.

Syntax How to Issue Application Namespace Maintenance Commands

```
EDAEXEC APP 'cmdname parameters'
```

Creating an Application Area

The APP CREATE command creates an application area under the approot location. Any number of applications can be created with one command. For UNIX, Windows NT/2000, OpenVMS, and OS/400, this application area is a directory in which all types of static application components are stored. On MVS, it is a number of DSNs that are created with a common root. These DSNs include:

- approot.appname.MASTER.DATA
- approot.appname.FOCEXEC.DATA
- approot.appname.ACCESS.DATA
- approot.appname.FOCSTYLE.DATA
- approot.appname.GIF.DATA
- approot.appname.HTML.DATA

Note: The word HOLD cannot be used as an application name.

Syntax How to Create an Application Area

```
APP CREATE appname1 appname2 appname3 [-]
          [appname4 ...]
```

where:

appname

Are application names. If you need to specify more appnames than can fit on one line, use the continuation character '-' and code more appnames on the next line.

Deleting an Application Area

The APP DELETE command deletes an application area. For UNIX, Windows NT/2000, OpenVMS, and OS/400, the directory under approot is deleted. For MVS, the application DSNs are deleted. On all platforms, the application area does not have to be empty. Any number of applications can be deleted with one command.

Syntax How to Delete an Application Area

```
APP DELETE appname1 appname2 appname3 [-]
          [appname4 ...]
```

where:

appname

Are application names. If you need to specify more appnames than can fit on one line, use the continuation character '-' and code more appnames on the next line.

Copying an Application Area

The APP COPY command copies the contents of application area 1 to application area 2. For UNIX, Windows NT/2000, OpenVMS, and OS/400, the APP COPY command copies the appname1 directory contents to the appname2 directory. For MVS, the appname1 DSN contents are copied to the appname2 DSNs.

Note: On MVS, the server must be APF authorized to use this command.

Syntax How to Copy an Application Area

```
APP COPY appname1 appname2
```

where:

appname1

Is the application being copied.

appname2

Is the application to which the contents of the first application are being copied.

Copying an Application Component

The APP COPYF command copies a single component from one application area to another. Optionally, the component can be renamed in the process.

Syntax How to Copy an Application Component

```
APP COPYF appname1 filename1 filetype1 appname2 filename2 filetype2
```

where:

appname1

Is the application name of the component being copied.

filename1

Is the component name to be copied.

filetype1

Is the component type of the component name to be copied.

appname2

Is the application name to which the component of the first application is being copied.

filename2

Is the component name, in the second application, after the copy process.

filetype2

Is the component type, in the second application, after the copy process.

Moving an Application Component

The APP MOVEF command moves a single component from one application area to another. Optionally, the component can be renamed in the process.

Syntax How to Move an Application Component

```
APP MOVEF appname1 filename1 filetype1 appname2 filename2 filetype2
```

where:

appname1

Is the application name of the component being moved.

filename1

Is the component name to be moved.

filetype1

Is the component type of the component name to be moved.

appname2

Is the application name to which the component of the first application is being moved.

filename2

Is the component name, in the second application, after the move process.

filetype2

Is the component type, in the second application, after the move process.

Renaming an Application Area

The APP RENAME command renames a complete application area.

Syntax How to Rename an Application Area

```
APP RENAME appname1 appname2
```

where:

appname1

Is the application name to be renamed.

appname2

Is the new application name.

Renaming an Application Component

The APP RENAMEF command renames a single component in an application area.

Note: On MVS, the server must be APF authorized to use this command.

Syntax How to Rename an Application Component

```
APP RENAMEF appname filename1 filename2 filetype
```

where:

appname

Is the application name of the component being renamed.

filename1

Is the component name to be renamed.

filename2

Is the new component name.

filetype

Is the component type of the filename to be renamed.

Deleting an Application Component

The APP DELETEF command deletes a single component in an application area.

Syntax How to Delete an Application Component

```
APP DELETEF appname filename filetype
```

where:

appname1

Is the application name of the component being deleted.

filename1

Is the component name to be deleted.

filetype

Is the component type of the filename to be deleted.

Listing an Application

The APP LIST command lists the applications available under approot.

Syntax How to List an Application

```
APP LIST [HOLD]
```

If the HOLD option is used, the output is written to a temporary file, called focappl.ftm, which can then be used in a report request. If the HOLD option is used, you can use the following request to view the output:

```
SELECT DATE,TIME,APPNAME FROM FOCAPPL
```

The APP LIST output is:

```
15/02/2000  13.36.38   baseapp
15/02/2000  13.36.38   ggdemo
15/02/2000  13.36.38   ncp
15/02/2000  13.36.38   template
```

Listing Components

The APP QUERY command lists all components for application appname. Any number of applications can be queried with one command.

Syntax How to List Components

```
APP QUERY appname1 appname2 appname3 [-]
         [appname4 ...] [HOLD]
```

where:

appname

Are application names. If you need to specify more appnames than can fit on one line, use the continuation character '-' and code more appnames on the next line.

If the HOLD option is used, the output is written to a temporary file, called focappq.ftm, which can then be used in a report request. If the HOLD option is used, you can use the following request to view the output:

```
SELECT DATE,TIME,SIZE,FILENAME,APPNAME FROM FOCAPPQ
```

The APP QUERY output is:

```
08/03/2000  13.49.32   118949        eas.mas    APP1
08/03/2000  13.49.32   164           eas.acx    APP1
```

Note: Only one appname can be used on MVS.

Listing Active Applications

THE APP SHOWPATH command lists all the currently active applications (including bareapp which is always last).

Syntax How to list Active Applications

```
APP SHOWPATH
```

The APP SHOWPATH output is:

```
ibisamp
bareapp
```

Providing Help Information

The APP HELP command provides help information for all of the APP commands.

Reference APP HELP Output

The following APP commands are available:

Command	Description
`* APP ENABLE`	Enables the app search, disables EDAPATH, EDASYN, and EDASYNR.
`* APP CREATE APP1 [APP2]`	Creates an application directory under apps without making it current.
`* APP DELETE APP1 [APP2]`	Deletes the directory.
`* APP COPY APP1 APP2`	Copies all files from app1 to app2 directory.
`* APP COPYF APP1 FNAME FILETYPE APP2 FNAME2 FILETYPE2`	Copies the component fname filetype from app1 to app2 as fname2 filetype2.
`* APP MOVEF APP1 FNAME FILETYPE APP2 FNAME2 FILETYPE2`	Moves the component fname filetype from app1 to app2 as fname2 filetype2.
`* APP RENAME APP1 APP2`	Renames app1 to app2.
`* APP RENAMEF APP1 FNAME FNAMENEW FILETYPE`	Renames the component fname filetype to fnamenew in the same application.
`* APP DELETEF APP1 FNAME FILETYPE`	Deletes the component fname filetype from app1.

Command	Description
`* APP PATH APP1 [APP2 ...]`	Sets APP1 [and APP2 ... - unlimited application directories] as an app search path, prior to permanent baseapp.
`* APP PATH`	Unsets the current app path.
`* APP APPENDPATH`	APP1 [APP2...] – append APP1 [APP2] to APP PATH.
`* APP PREPENDPATH`	APP1 [APP2...] – prepend APP1 [APP2] to APP PATH.
`* APP HOLD APP1`	Defines directory to hold temporary files as APP1.
`* APP LIST`	Lists all applications that exist (not necessarily in APP PATH).
`* APP LIST HOLD`	Same to save in the file focappl.ftm in the TEMP directory.
`* APP QUERY APP1 [APP2...]`	Provides an annotated listing of files in the application.
`* APP QUERY APP1 HOLD`	Same to save in focappq.ftm in the TEMP directory.
`* APP FI CAR DISK APP1/ validcar.dat`	Allocates a sequential file in a uniform syntax (UNIX, Windows NT/2000, OpenVMS, and OS/400) in the directory corresponding to the APP1.
`* APP HELP`	Lists the above help.

Application Namespace Run-Time Commands

When the Application Namespace feature is enabled, the search path for reading components is changed. For details, see *Affecting the Search Path for Application Components* on page 8-14.

Note: The run-time commands can be ordered in any supported server profile.

Syntax How to Enable Application Namespace

To enable Application Namespace, use the following command:

```
APP ENABLE
```

On UNIX, Windows NT/2000, OpenVMS, and OS/400, the logical names EDAPATH, EDASYN, EDASYNR, and HOLDMAST are disabled and cause an error condition if they are set during the session. Also, the current directory is excluded from the search path.

Note: To disable the Application Namespace feature, remove the APP ENABLE command and any other run-time APP commands from the server.

Affecting the Search Path for Application Components

The APP PATH command affects the search path for application components.

On UNIX, Windows NT/2000, OpenVMS, and OS/400, the appnames are single level directories under the approot directory and were created with the APP CREATE command. Multiple appnames can be specified to extend the search path. When APP PATH is used, the following search path is in effect:

- Directories specified in the APP HOLD command (if not issued, then EDATEMP directory is searched).
- Directories under approot specified in the APP PATH command.
- Default directory baseapp.
- The catalog directory under EDAHOME.

Note: If APP Path is issued without an appname, the current APP Path setting is removed from the search path.

On MVS, the appname is used as a qualifier to be appended to the approot value in a dynamic allocation command when searching for a component DSN.

Syntax How to Affect the Search Path for Application Components

```
APP PATH appname1 appname2 appname3 [-]
        [appname4 ...]
```

where:

appname

Are application names. If you need to specify more appnames than can fit on one line, use the continuation character '-' and code more appnames on the next line.

Reference Search Path for APP Commands

The following chart indicates the appropriate search paths for the following APP commands:

- APP ENABLE
- APP PATH APP1

Platform	Approot setting	Search path (No APP HOLD)
OS/390 and z/OS, OS/400	ibi/apps	EDATEMP agent directory ibi/apps/app1 ibi/apps/baseapp $EDAHOME/catalog
Windows NT/2000	ibi\apps	EDATEMP agent directory ibi\apps\app1 ibi\apps\baseapp %EDAHOME%\catalog
MVS	EDAARH.MYAPPS	Master Files [Temporary hold area for session] EDAARH.MYAPPS.APP1.MASTER.DATA EDAARH.MYAPPS.BASEAPP.MASTER.DATA ddname IBIMFD Access Files EDAARH.MYAPPS.APP1.ACCESS.DATA EDAARH.MYAPPS.BASEAPP.ACCESS.DATA ddname IBIAFD FOCEXEC Files (RPCs) EDAARH.MYAPPS.APP1.FOCEXEC.DATA EDAARH.MYAPPS.BASEAPP.FOCEXEC.DATA ddname IBIFEX

Platform	Approot setting	Search path (No APP HOLD)
OpenVMS	disk$dua0: [EDA.IBI.APPS]	EDATEMP agent directory disk$dua0:[EDA.IBI.APPS.APP1] disk$dua0:[EDA.IBI.APPS.BASEAPP] EDAHOME [.CATALOG]

The chart below indicates the search paths for the following APP commands:

- APP ENABLE
- APP PATH APP1
- APP HOLD APP1

Platform	Approot setting	Search path
OS/390 and z /OS,OS/400	ibi/apps	ibi/apps/app1 ibi/apps/baseapp $EDAHOME/catalog
Windows NT/2000	ibi\apps	ibi\apps\app1 ibi\apps\baseapp %EDAHOME%\catalog
MVS	EDAARH.MYAPPS	Master Files EDAARH.MYAPPS.APP1.MASTER.DATA EDAARH.MYAPPS.BASEAPP.MASTER.DATA ddname IBIMFD Access Files EDAARH.MYAPPS.APP1.ACCESS.DATA EDAARH.MYAPPS.BASEAPP.ACCESS.DATA ddname IBIAFD FOCEXEC Files (RPCs) EDAARH.MYAPPS.APP1.FOCEXEC.DATA EDAARH.MYAPPS.BASEAPP.FOCEXEC.DATA ddname IBIFEX
OpenVMS	disk$dua0: [EDA.IBI.APPS]	disk$dua0:[EDA.IBI.APPS.APP1] disk$dua0:[EDA.IBI.APPS.BASEAPP] EDAHOME [.CATALOG]

Adding Application Names to the Beginning of a Search Path

The APP PREPENDPATH command allows application names to be added to the beginning of an existing APP PATH search path.

Syntax How to Add Application Names to the Beginning of the Existing Path

```
APP PREPENDPATH appname1 appname2 appname3 [-]
               [appname4 ...]
```

where:

appname

Are application names. If you need to specify more appnames than can fit on one line, use the continuation character '-' and code more appnames on the next line.

Adding Application Names to the End of a Search Path

The APP APPENDPATH command allows application names to be added to the end of an existing APP PATH search path.

Syntax How to Add Application Names to the End of the Existing Path

```
APP APPENDPATH appname1 appname2 appname3 [-]
              [appname4 ...]
```

where:

appname

Are application names. If you need to specify more appnames than can fit on one line, use the continuation character '-' and code more appnames on the next line.

Controlling Where the Output File Is Created

The APP HOLD command controls where the output file is created for any write process in the application if an APP FI command has not been issued. This includes:

`CREATE SYNONYM`

Is not affected by the APP FI command. It is not recommended to use the APP HOLD command in conjunction with the CREATE SYNONYM command. The recommendation for CREATE SYNONYM is

```
CREATE SYNONYM appname/synonymname
```

`ON TABLE HOLD`

Should be used with caution and the APP HOLD command should be turned off immediately following the process that creates the output file. To turn off the effects of the command, issue APP HOLD without an appname.

The APP HOLD command is intended to be used to refresh files that are common across all users of the application. It should not be used in the application for private files as it points to an application area that is used by multiple users. If the same hold name (HOLD or AS name for example) is used, conflicts between users could result.

Syntax How to Control Where the Output File Is Created

```
APP HOLD appname
```

where:

appname

Is an application name.

Specifying Physical File Location

This APP FI command provides a platform-independent mechanism for allocating data files.

On UNIX, Windows NT/2000, OpenVMS, and OS/400, the physical name is the actual name as stored in the approot/appname directory.

On MVS, the DSN is:

```
approot.appname.physical_name
```

Syntax How to Specify Physical File Location

```
APP FI filename DISK appname/physical_name
```

where:

filename

Is the logical file name as known by the server.

appname

Is an application name.

physical_name

Is the file name stored in the application area.

Example Sample Physical File Location on MVS

```
APPROOT=EDAARH.MYAPPS
APP FI CAR APP1/CAR.FOCUS
```

The DSN that is dynamically allocated is

```
EDAARH.MYAPPS.APP1.CAR.FOCUS
```

Note: On MVS, this file must already exist.

Syntax How to Specify Physical File Location Outside of APPROOT

To assist with existing applications outside the pre-defined application root directory (APPROOT), the APP MAP command allows an alias to be assigned to a non-APPROOT directory. This alias becomes a virtual directory under APPROOT so it can then be referenced in an APP PATH command. Mapping does NOT automatically add to the path; it simply makes it available to participate in an APP PATH command.

```
APP MAP virtualname realdirectory
```

where:

virtualname

Is an 8-character application name that can later be used in an APP PATH command.

realdirectory

Is a real full path directory name in the native style of the given operation system.

Note: On OS/400, the APP MAP command can only be used to map an IFS directory and not a QSYS library.

Mapping DDNAME Allocations

On OS/390 and z/OS, the APP MAP command can be used to map to ddname allocations for different components of an application. These ddnames can be in the EDASTART JCL or DYNAM'ed allocations.

Syntax How to Map DDNAME Allocations

The syntax of the APP MAP command for this type of mapping is as follows

```
APP MAP applicationname file_extension=//dd:ddname;file_extension=//dd:ddname; ...
```

where:

applicationname

Is the name to be used, in the APP PATH or APP APPENDPATH commands, to reference this mapping.

file_extension

Is one of the following valid server file extensions

```
.mas
.fex
.acx
.htm
```

ddname

Is the ddname of the allocation you wish to map to.

Example Using the APP MAP Command to Map DDNAME Allocations

```
DYNAM ALLOC FILE MYMAS DA EDAARH.MASTER.DATA SHR REU
APP MAP APP1 MAS=//DD:MYMAS;
APP APPENDPATH APP1
```

The server, by default, will have an APP MAP command in the edasprof.prf file to map application name MVSAPP to allocations MASTER, FOCEXEC, ACCESS and HTML.

Simulating the APP MAP Command on MVS

The APP MAP command is not supported on MVS. To accomplish the mapping feature, ALIAS names can be created to point to Datasets that are required to participate as application name spaces.

Example Creating an APROOT Style Alias Pointing to an Existing Data Set Name

The following JCL, using an APPROOT value of EDAARH.MYAPPS can be used as an example:

```
//IDCAMS EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 DELETE EDAARH.MYAPPS.APPLEG1.MASTER.DATA ALIAS
 DEFINE ALIAS (NAME(EDAARH.MYAPPS.APPLEG1.MASTER.DATA)-
    RELATE(EDAARH.MASTER.DATA))
/*
```

The above JCL example creates an APROOT style alias pointing to an existing DSN, which contains Master Files. You can create any combination of ALIAS names to match the six datasets that are required to complete an application name space. See *How to Create an Application Area* on page 8-6.

The APP LIST command returns APPLEG1 as a valid application name space, APP QUERY can be used to list APPLEG1 components. The APP PATH must be used to make APPLEG1 available to the client application.

Executing a FOCEXEC Outside of APPROOT

To assist with application isolation and avoiding constant manual set and reset of application paths, an automatic and dynamic set/reset feature has been added to the EX syntax that also supports application specific profiles. Effectively it adds the target to the beginning of the APP PATH and then removes it after execution of the FOCEXEC and all FOCEXECs that the original one calls.

Syntax How to Execute a FOCEXEC Outside of APPROOT

The behavior is activated with the following syntax:

```
EX MYAPP/myfex[.ext]
```

where:

`MYAPP`

Is an application directory under the APPROOT (or an APP MAP directory).

`myfex``[.ext]`

Is a FOCEXEC in the MYAPP directory with an extension of .fex. The .ext option can be used to specify an alternate file extension.

If the application also contains a profile.fex in the application directory MYAPP, it will be executed before the actual requested FOCEXEC.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Reader Comments

In an ongoing effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. You can contact us through the following methods:

Mail: Documentation Services - Customer Support
Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898

Fax: (212) 967-0460

E-mail: books_info@ibi.com

Web form: http://www.informationbuilders.com/bookstore/derf.html

Name:_____

Company:_____

Address:_____

Telephone:_____Date:_____

E-mail:_____

Comments:

Reader Comments

Information Builders, Two Penn Plaza, New York, NY 10121-2898 **(212) 736-4433**